State of the STARK

Eli Ben-Sasson, Chief Scientist (East) | October 2018

Overview

What's a STARK?

STARK comparison to

- SNARKs (Zcash)
- Recursive SNARKs (Coda)
- Bulletproofs (Monero)

Benefits of STARK for scalability

Potential use-cases

Scalability, Privacy & Proofs

Proofs of Computational Integrity

INTEGRITY

The quality of being honest
(Dictionary)

Proofs of Computational Integrity

INTEGRITY

The quality of being honest
(Dictionary)

COMPUTATIONAL INTEGRITY

The quality of a computation
being executed honestly

Proofs of Computational Integrity

INTEGRITY

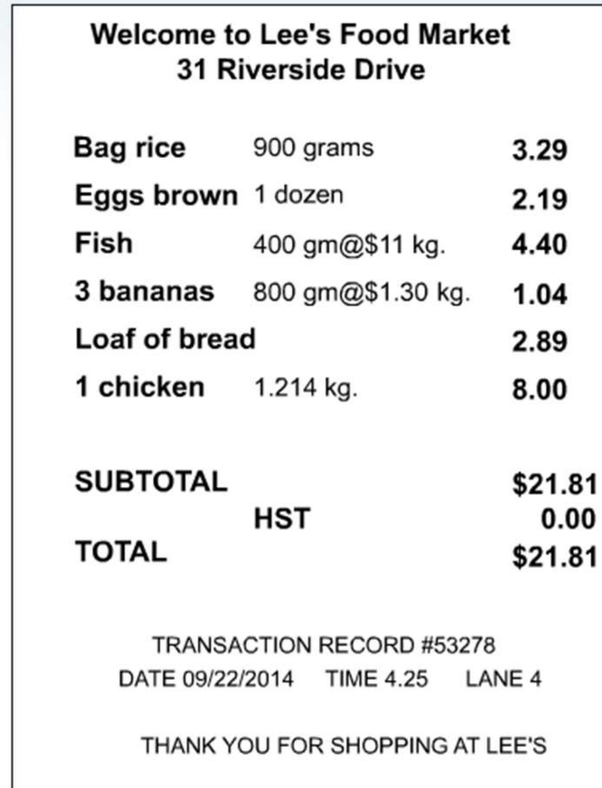
The quality of being honest
(Dictionary)

COMPUTATIONAL INTEGRITY

The quality of a computation being executed honestly

Grocery receipts are proofs of computational integrity

- Verification via naive re-execution of computation
- Proof is (i) deterministic, (ii) error free, (iii) one-shot (non-interactive)
- Modern CI proofs have (i) randomness, (ii) small error, (iii) interaction; in return, offer many benefits…


Welcome to Lee's Food Market
31 Riverside Drive

Bag rice	900 grams	3.29
Eggs brown	1 dozen	2.19
Fish	400 gm@$11 kg.	4.40
3 bananas	800 gm@$1.30 kg.	1.04
Loaf of bread		2.89
1 chicken	1.214 kg.	8.00
SUBTOTAL		$21.81
	HST	0.00
TOTAL		$21.81

TRANSACTION RECORD #53278
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S

Modern proofs of Computational Integrity

Invented by Goldwasser, Micali, Rackoff in 1985:

Welcome to Lee's Food Market
31 Riverside Drive

Bag rice	900 grams	**3.29**
Eggs brown	1 dozen	**2.19**
Fish	400 gm@$11 kg.	**4.40**
3 bananas	800 gm@$1.30 kg.	**1.04**
Loaf of bread		**2.89**
1 chicken	1.214 kg.	**8.00**
SUBTOTAL		**$21.81**
	HST	**0.00**
TOTAL		**$21.81**

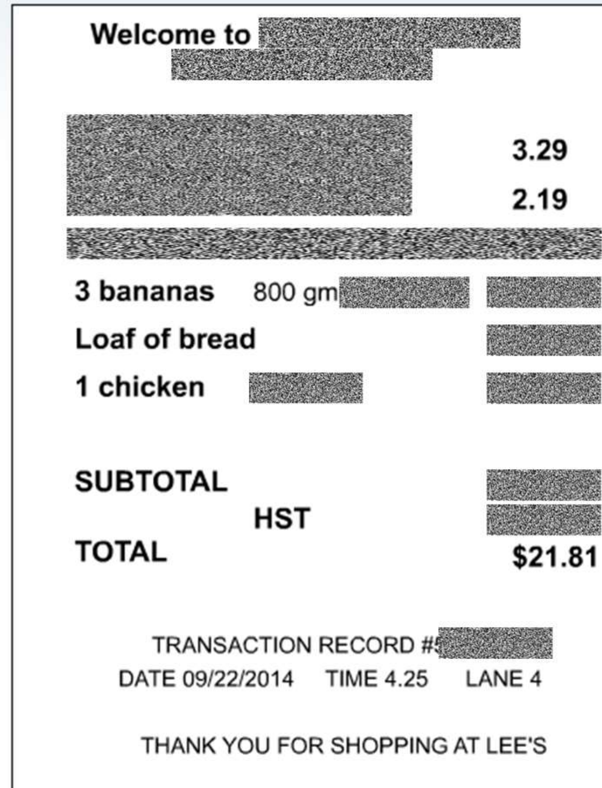
TRANSACTION RECORD #53278
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S

Modern proofs of Computational Integrity

Invented by Goldwasser, Micali, Rackoff in 1985:

- **Zero knowledge (ZK):** private inputs are shielded



Welcome to [redacted]

[redacted] 3.29
2.19
[redacted]

3 bananas 800 gm [redacted]
Loaf of bread [redacted]
1 chicken [redacted]

SUBTOTAL [redacted]
HST [redacted]
TOTAL $21.81

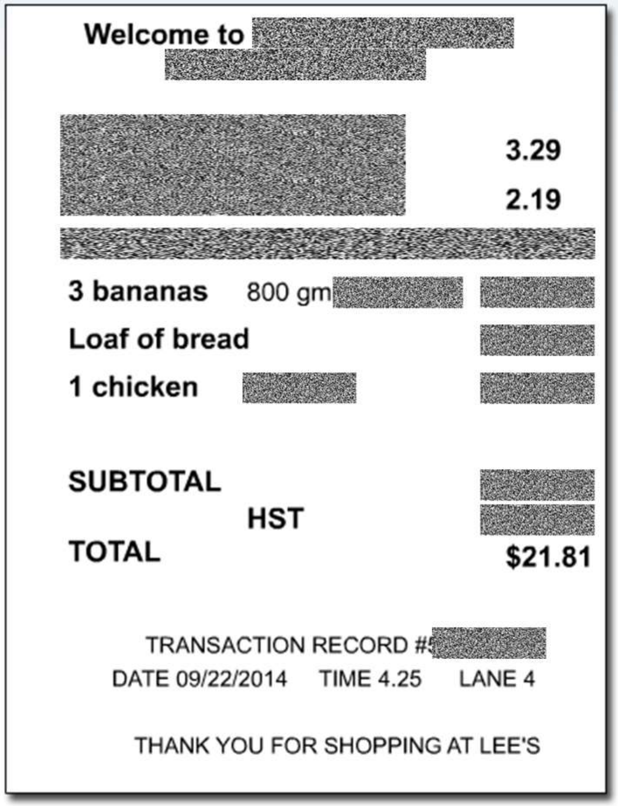
TRANSACTION RECORD #[redacted]
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S

Modern proofs of Computational Integrity

Invented by Goldwasser, Micali, Rackoff in 1985:

- **Zero knowledge (ZK):** private inputs are shielded
- **Scalability:** for computation lasting T cycles, proofs
  - generated in ~ T cycles (quasi-linear in T), and
  - verified exponentially faster than T (~ log T cycles)
- **Universality (Turing Completeness):** apply to any computation


Welcome to [redacted]

[redacted] 3.29
2.19
[redacted]

3 bananas	800 gm[redacted]	[redacted]
Loaf of bread		[redacted]
1 chicken	[redacted]	[redacted]
SUBTOTAL		[redacted]
	HST	[redacted]
TOTAL		$21.81

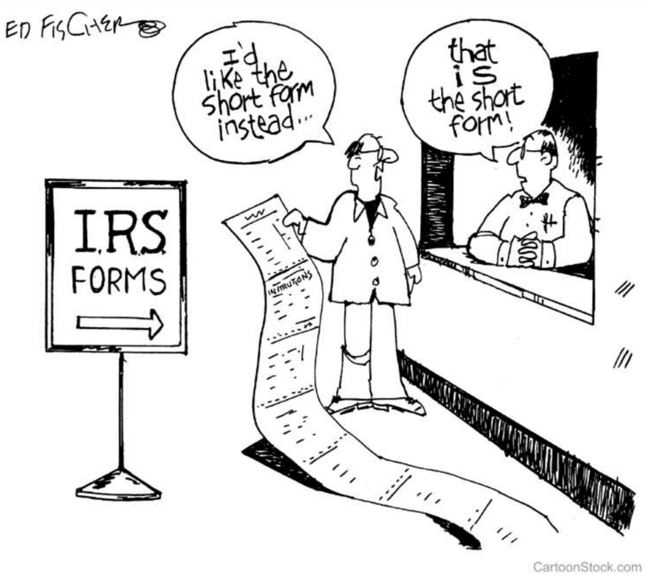
TRANSACTION RECORD #[redacted]
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S

Modern proofs of Computational Integrity

Invented by Goldwasser, Micali, Rackoff in 1985:

- **Zero knowledge (ZK):** private inputs are shielded
- **Scalability:** for computation lasting T cycles, proofs
  - generated in ~ T cycles (quasi-linear in T), and
  - verified exponentially faster than T (~ log T cycles)
- **Universality (Turing Completeness):** apply to any computation



Examples of things one can prove:

- **Paid taxes on all my cryptocurrency tx's for 2017**
- **Control at least 10 ETH as of today**
- **Crypto exchange is in the black as of today (pf of solvency)**
- …

STARKWARE

ZK-STARK: core attributes

Many flavors of proof systems

Variety of theoretical constructions (past 30 yrs)

PCP based, linear PCPs, elliptic curve+pairing based succinct NIZKs, quadratic span/arithmetic programs (QAP/QSP), interactive oracle proofs (IOP), …

STARKWARE

Many flavors of proof systems

Variety of theoretical constructions (past 30 yrs)

…and implementations (past 5 yrs)

See zkp.science

PCP based, linear PCPs, elliptic curve+pairing based succinct NIZKs, quadratic span/arithmetic programs (QAP/QSP), interactive oracle proofs (IOP), …

Pinocchio, libsnark, zcash, pepper, ligero, bulletproofs, libstark, aurora, …

zk-STARK definition

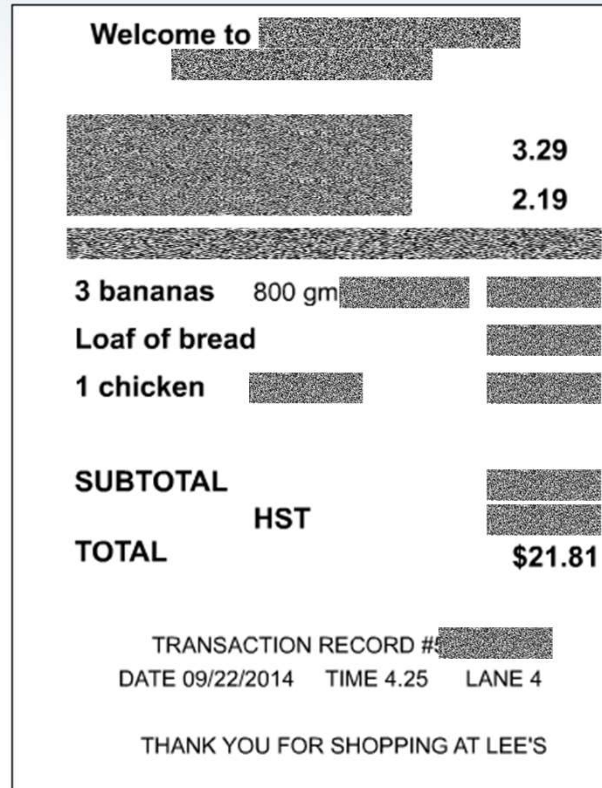
A proof system is a zk-STARK if it satisfies:

zk — **zero knowledge**: private inputs are shielded

S — **Scalable:** proofs for CI of computation lasting T cycles are
- generated in roughly T cycles (quasi-linear in T), and
- verified exponentially faster than T (roughly log T cycles)

T — **Transparent**: verifier messages are random coins; no trusted setup

AR K — **ARgument of Knowledge:** proof can be generated only by party knowing private input (formally: an efficient procedure can extract the secrets from a prover)


Welcome to [redacted]

[redacted]	3.29
	2.19
[redacted]	
3 bananas 800 gm [redacted]	[redacted]
Loaf of bread	[redacted]
1 chicken [redacted]	[redacted]
SUBTOTAL	[redacted]
HST	[redacted]
TOTAL	$21.81

TRANSACTION RECORD #[redacted]
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S

zk-STARK definition

A proof system is a zk-STARK if it satisfies:

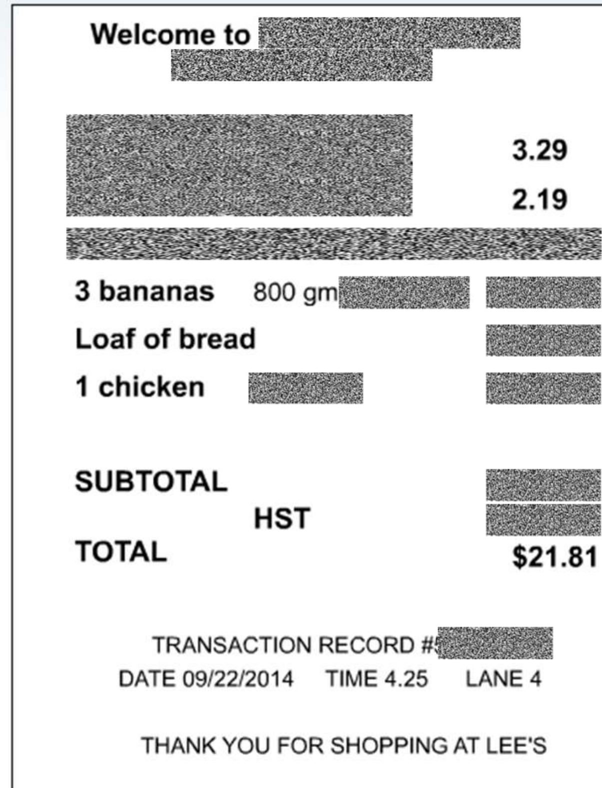
zk — **zero knowledge**: private inputs are shielded

S — **Scalable:** proofs for CI of computation lasting T cycles are
- generated in roughly T cycles (quasi-linear in T), and
- verified exponentially faster than T (roughly log T cycles)

T — **Transparent**: verifier messages are random coins; no trusted setup

AR K — **ARgument of Knowledge:** proof can be generated only by party knowing private input (formally: an efficient procedure can extract the secrets from a prover)

- STARKs may be interactive (use blockchain as source of transparent randomness), gives shorter & safer proofs
- 1st STARK: SCI-POC [BCG+16]; 1st zk-STARK: libstark [BBHR18]



Welcome to [redacted]

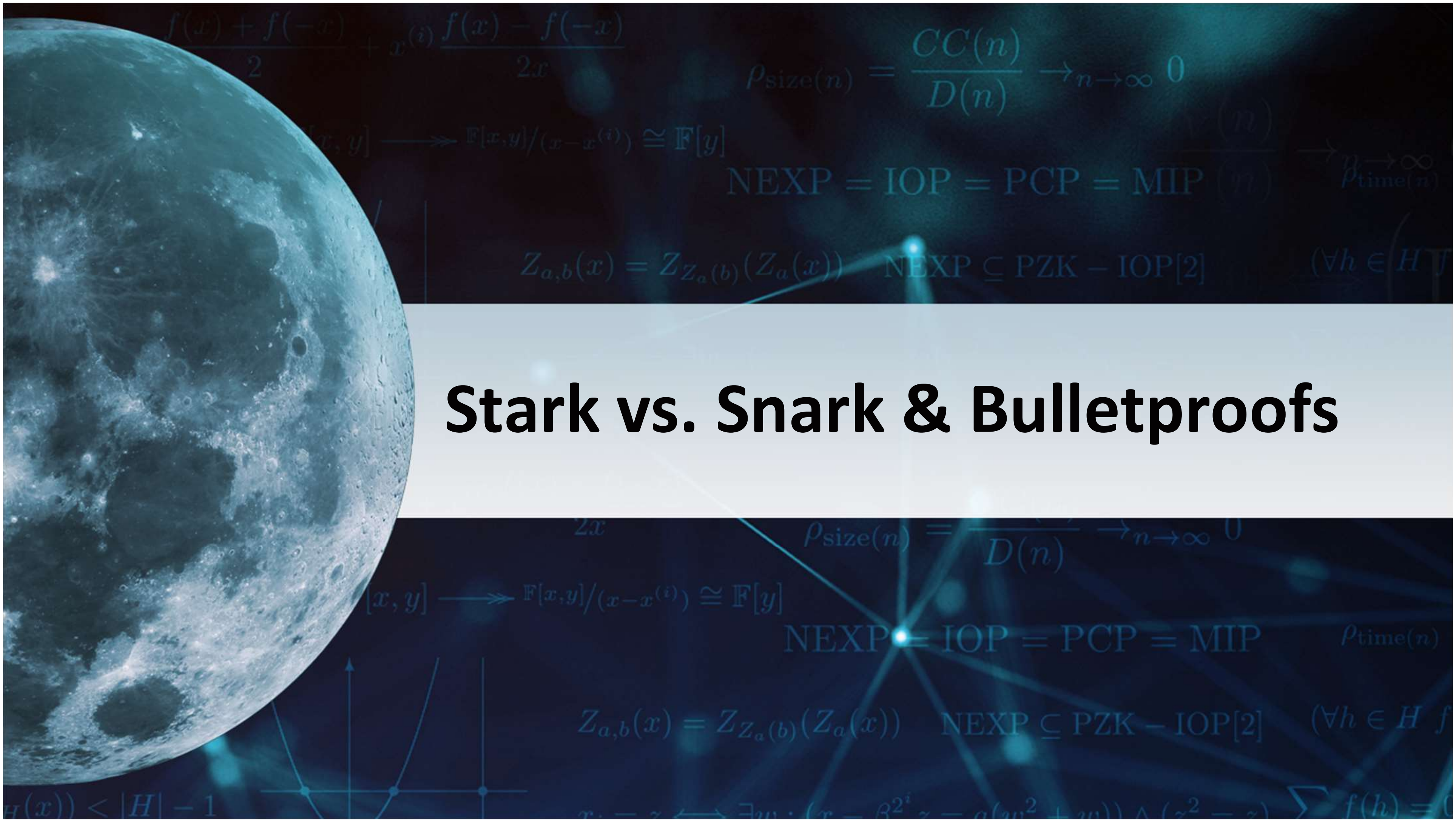
[redacted] 3.29
2.19
[redacted]

3 bananas 800 gm[redacted] [redacted]
Loaf of bread [redacted]
1 chicken [redacted] [redacted]

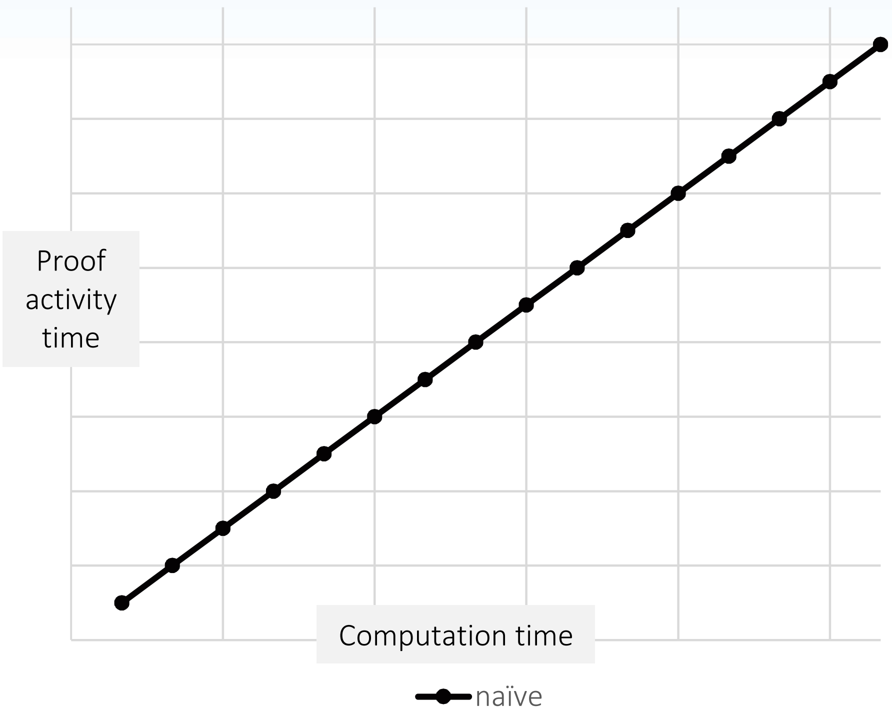
SUBTOTAL [redacted]
HST [redacted]
TOTAL $21.81

TRANSACTION RECORD #[redacted]
DATE 09/22/2014 TIME 4.25 LANE 4

THANK YOU FOR SHOPPING AT LEE'S


Stark vs. Snark & Bulletproofs

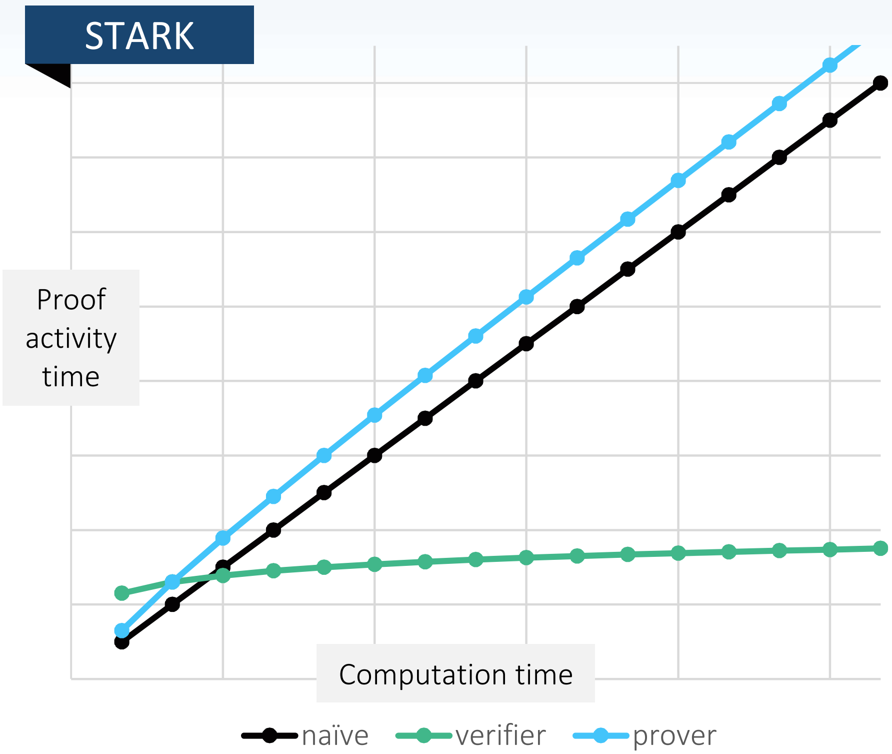
Scalability - Stark vs naive verification



Naive:

Verification = proving

Scalability - Stark vs naive verification



Naive:

Verification = proving

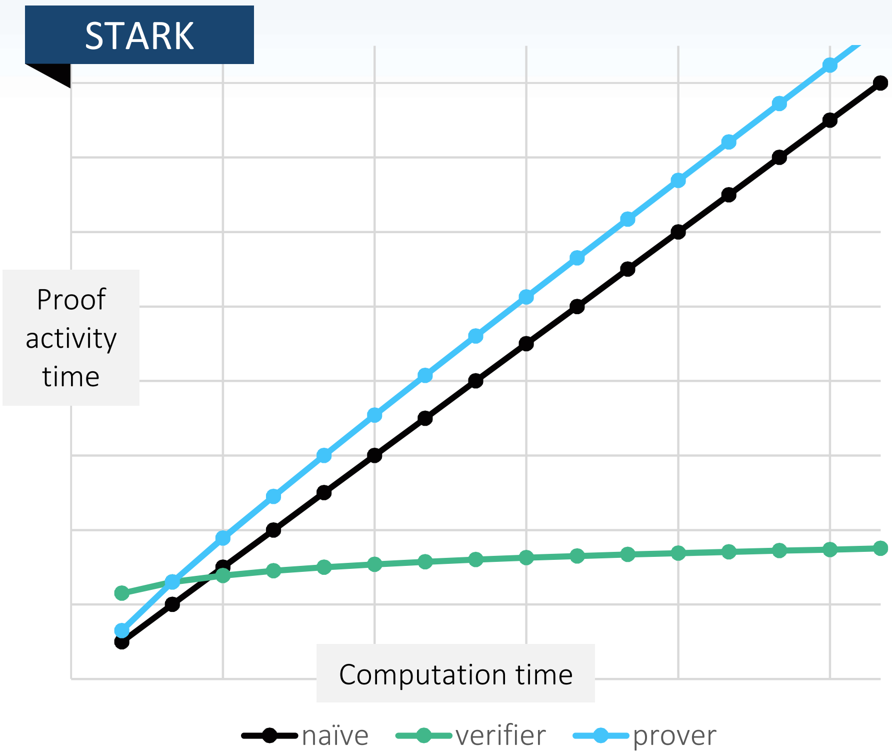
STARK:

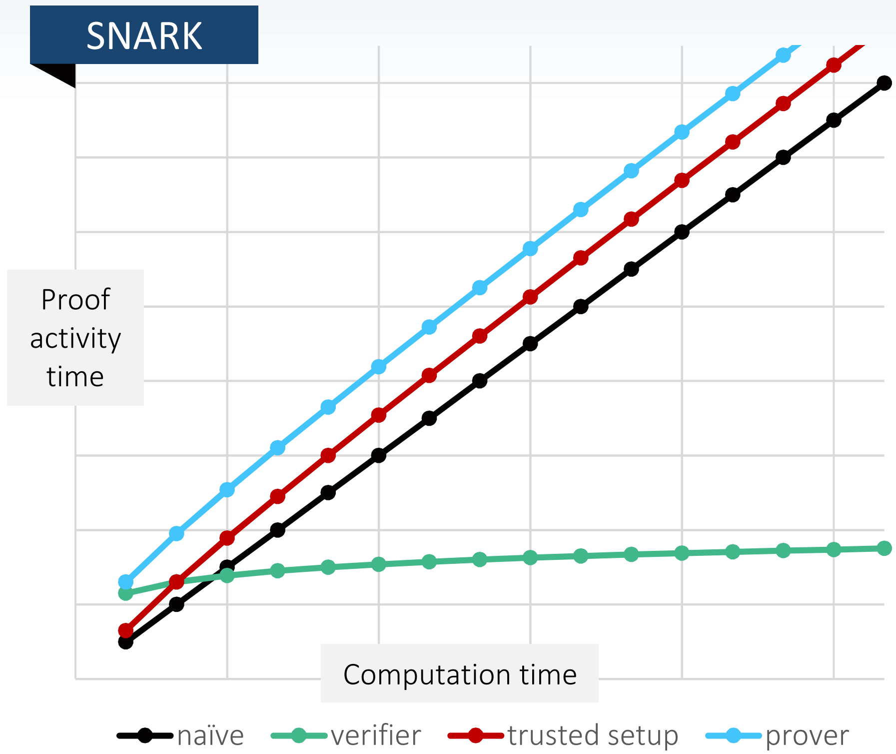
Quasi-linear proving

Poly-logarithmic verification

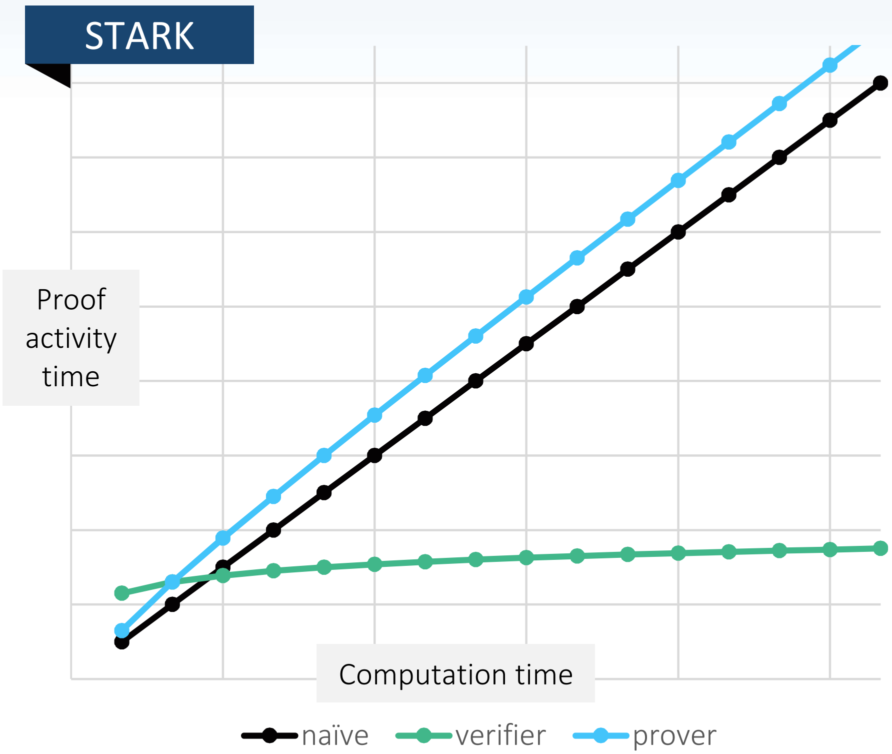
STARKWARE

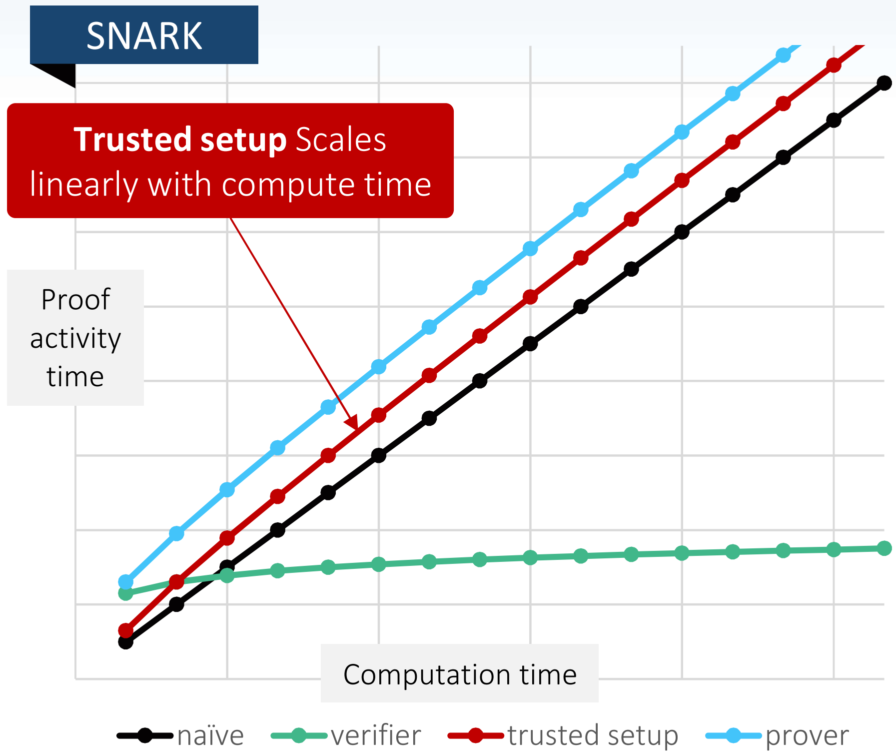
Scalability - Stark vs Snark (Zcash)





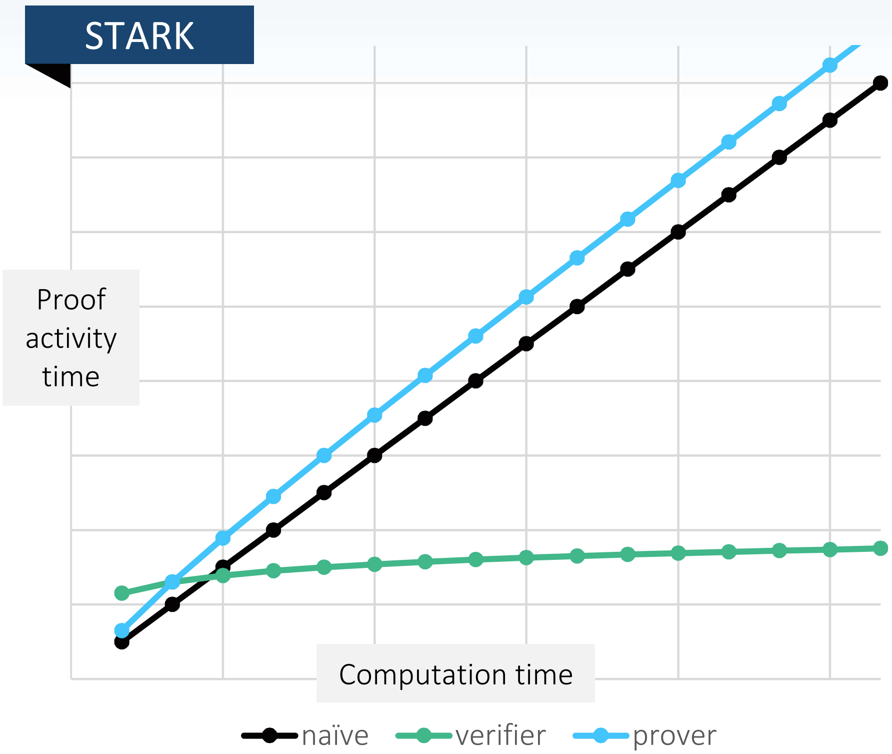
Scalability - Stark vs Snark (Zcash)

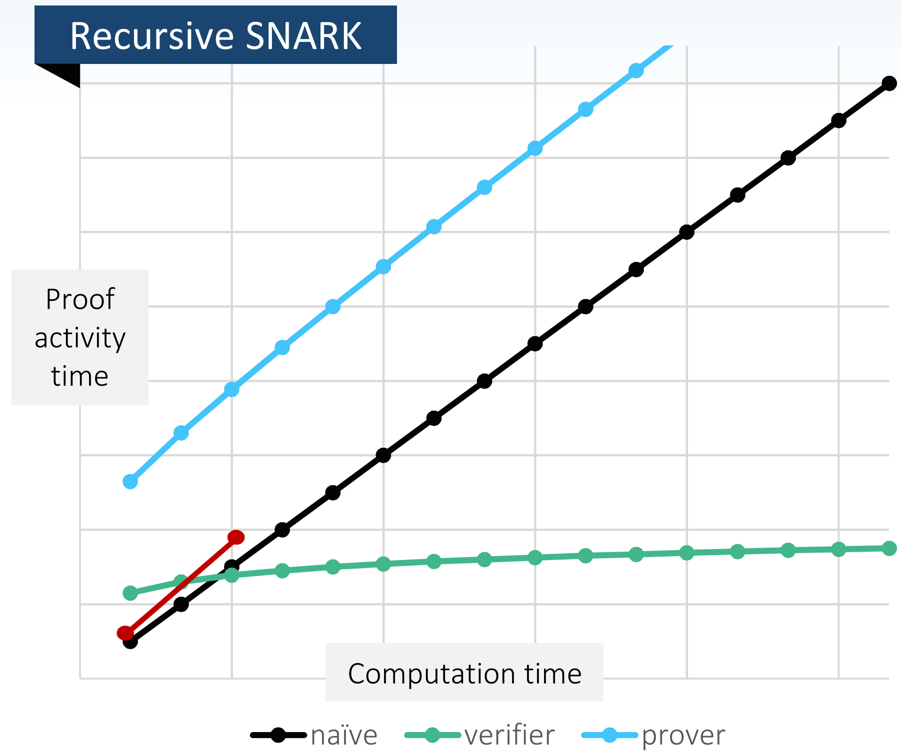




STARKWARE

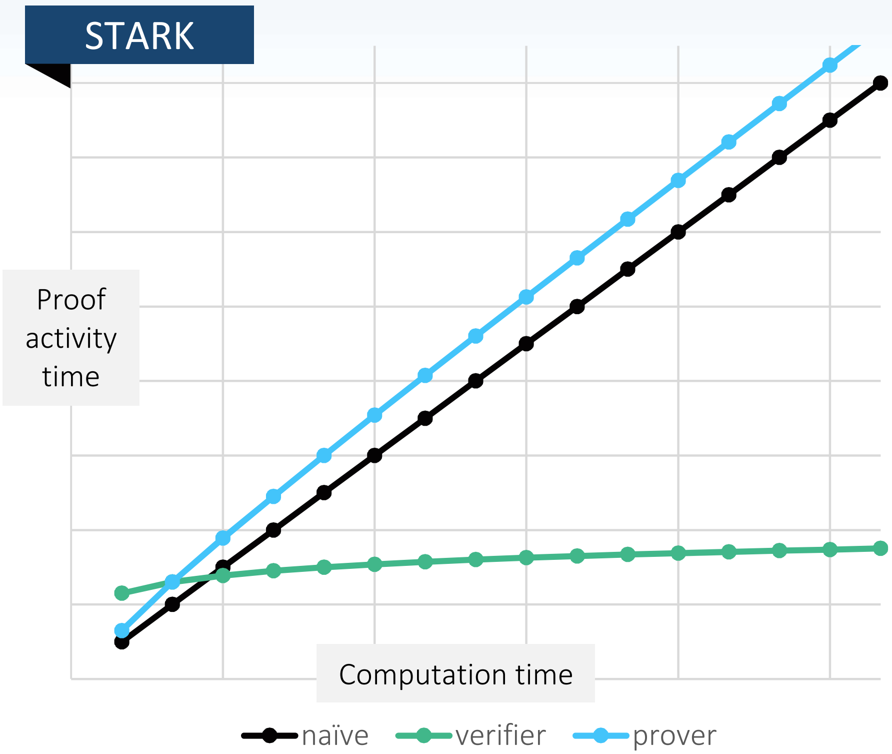
Scalability - Stark vs recursive Snark (Coda)

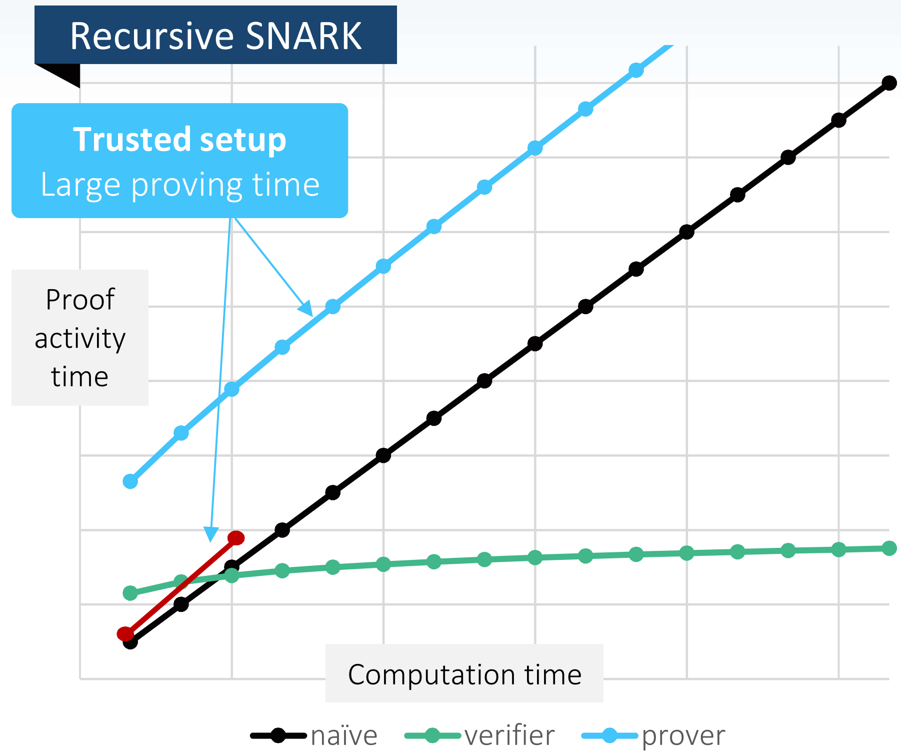




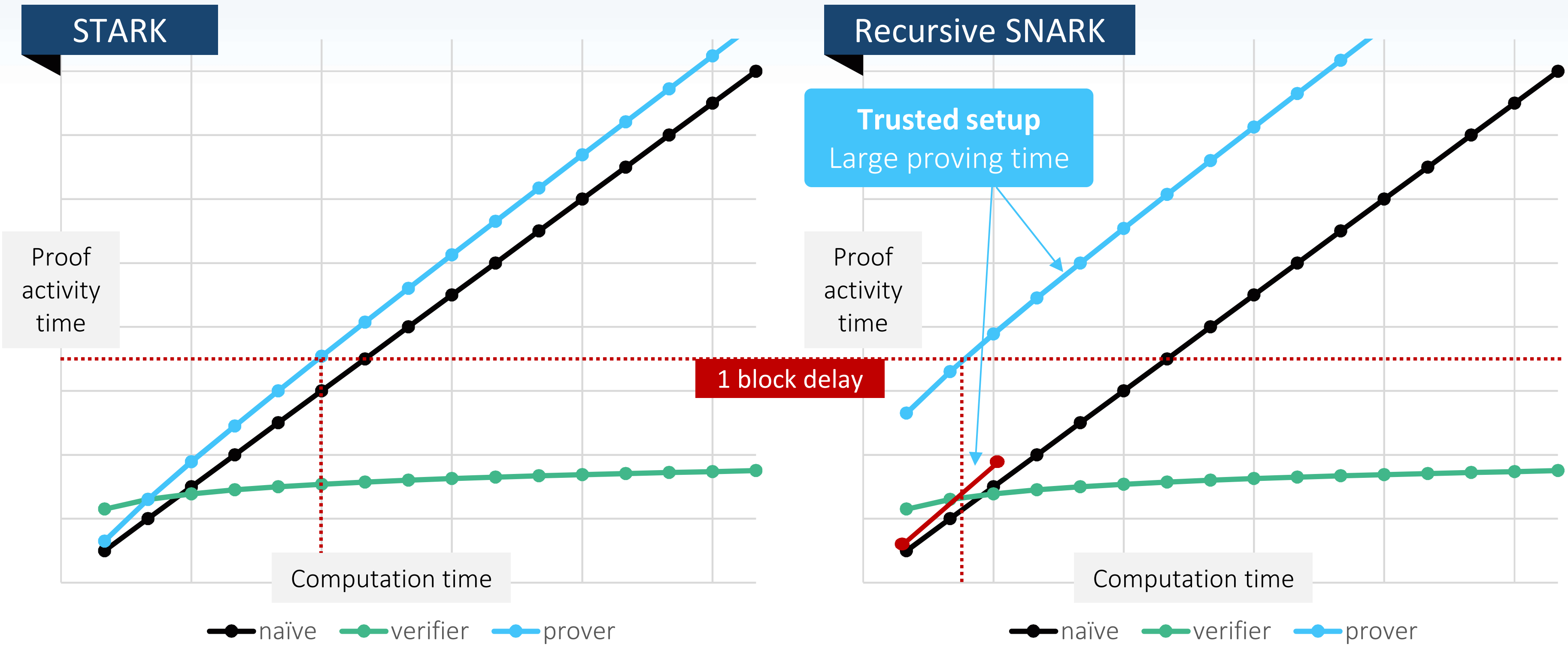
STARKWARE

Scalability - Stark vs recursive Snark (Coda)

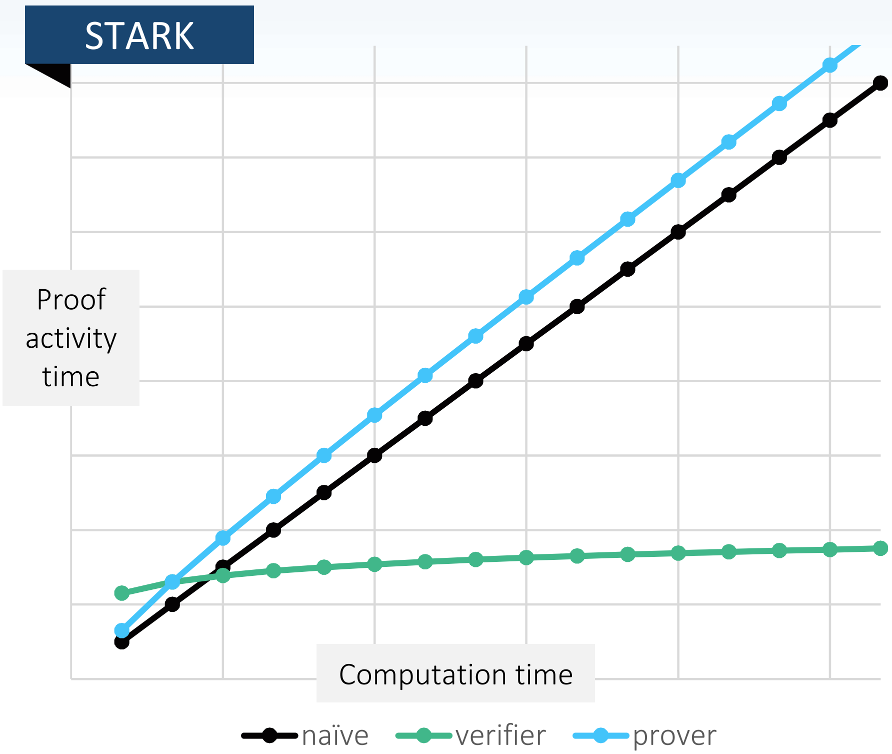


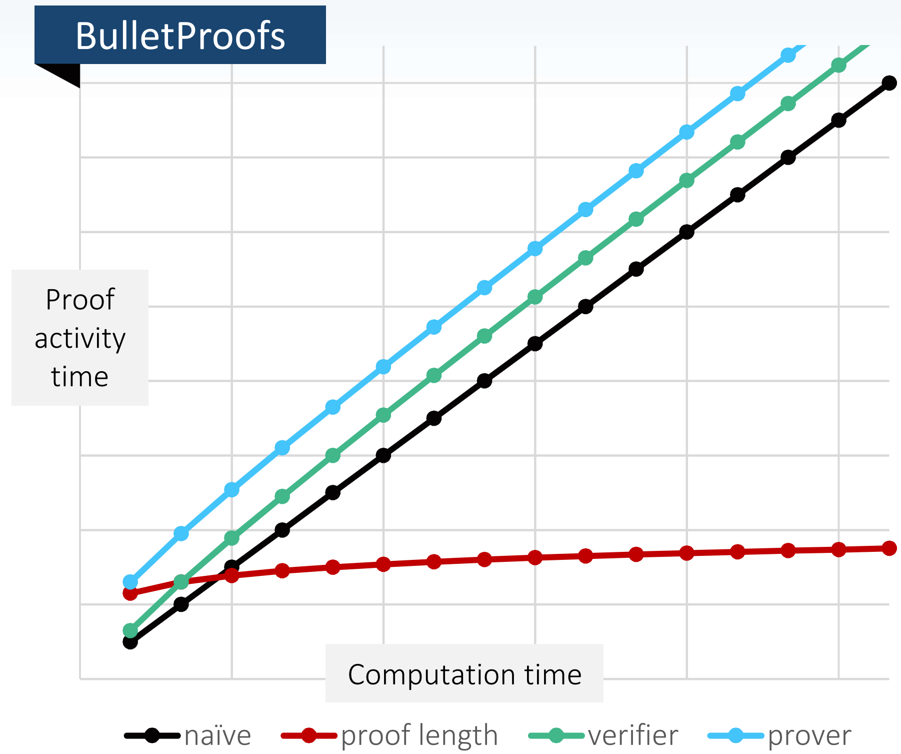


Scalability - Stark vs recursive Snark (Coda)

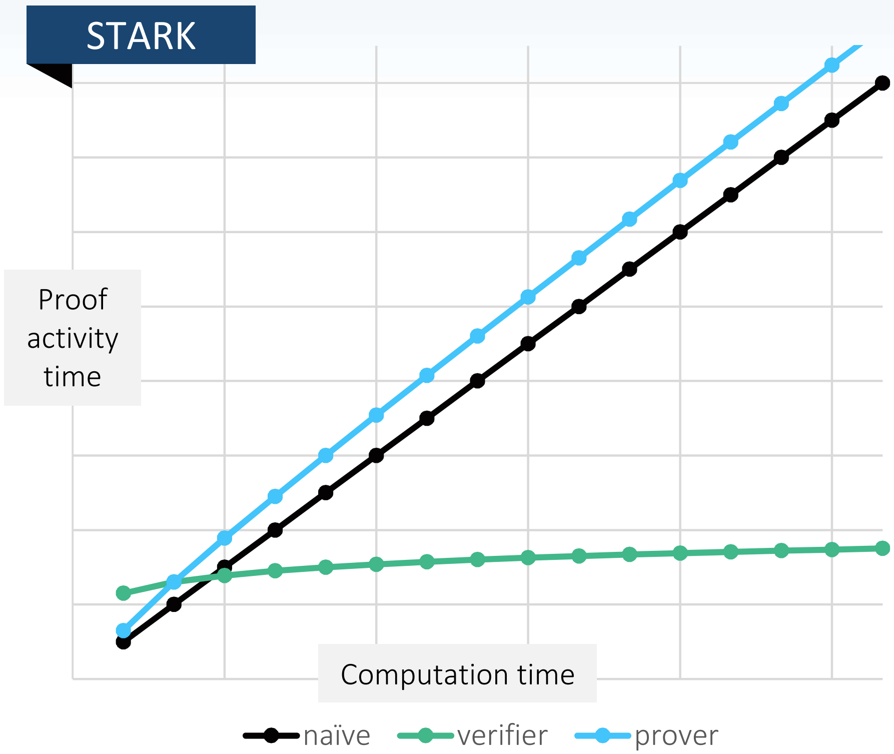


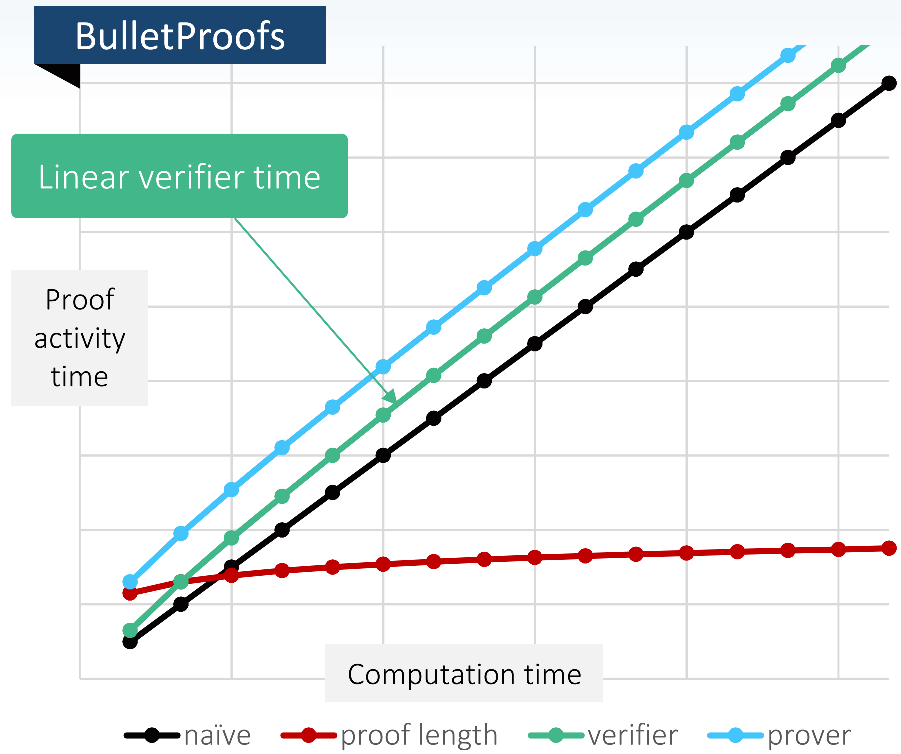
Stark vs BulletProofs (Monero)





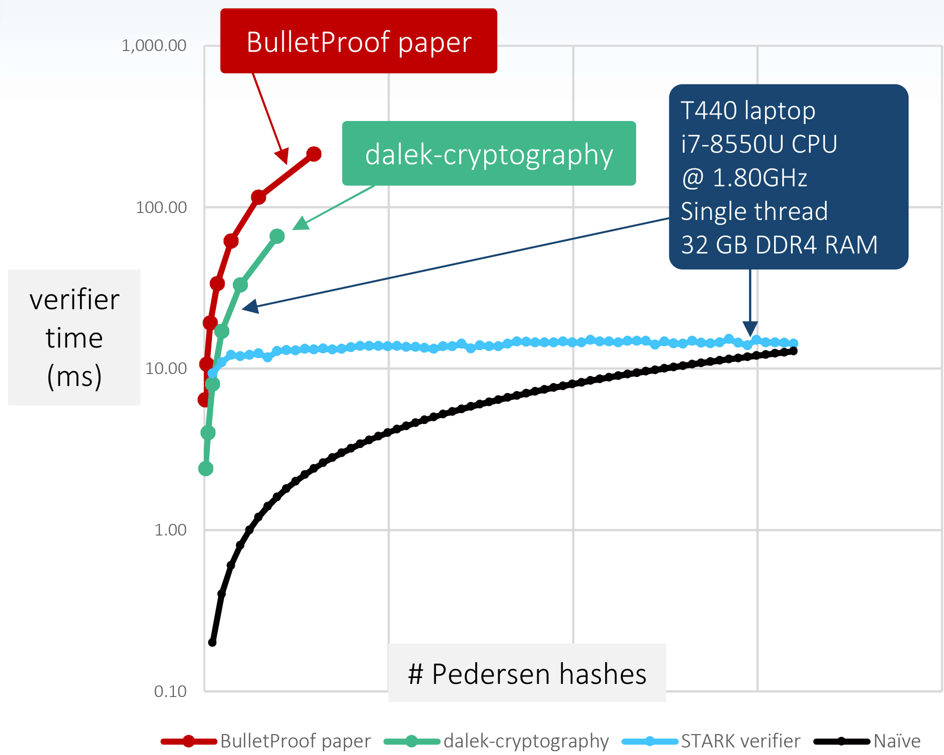
Stark vs BulletProofs (Monero)

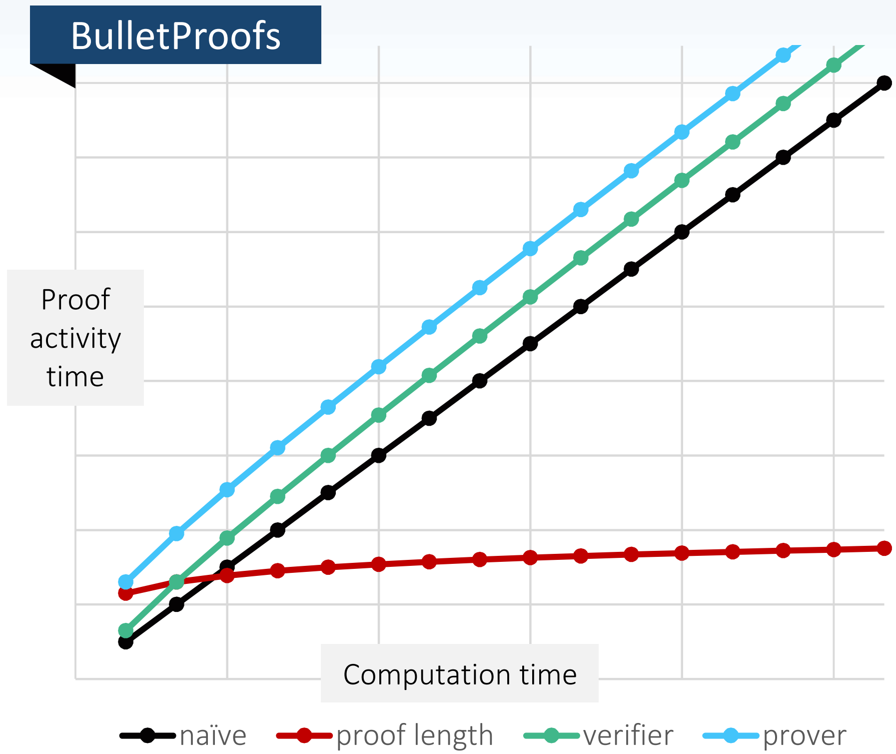




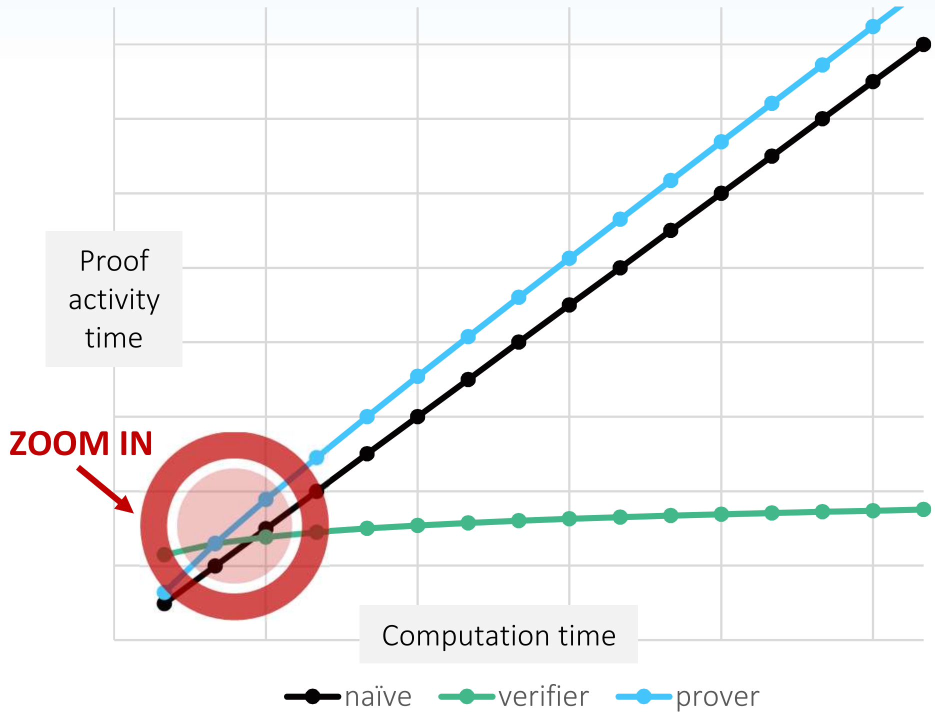
STARKWARE

STARK vs BulletProofs verification measurements



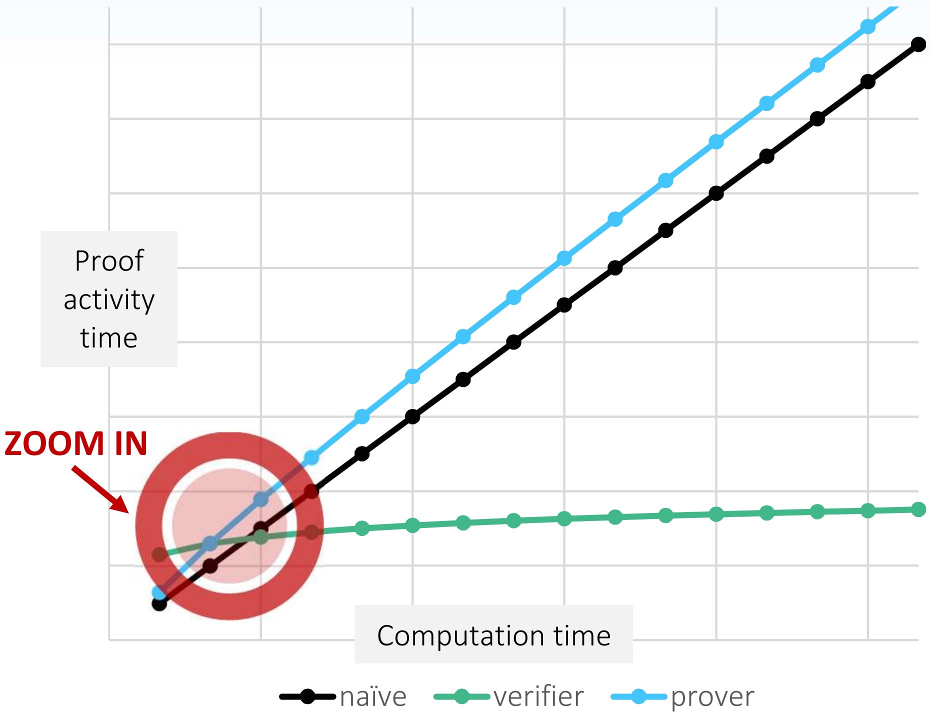


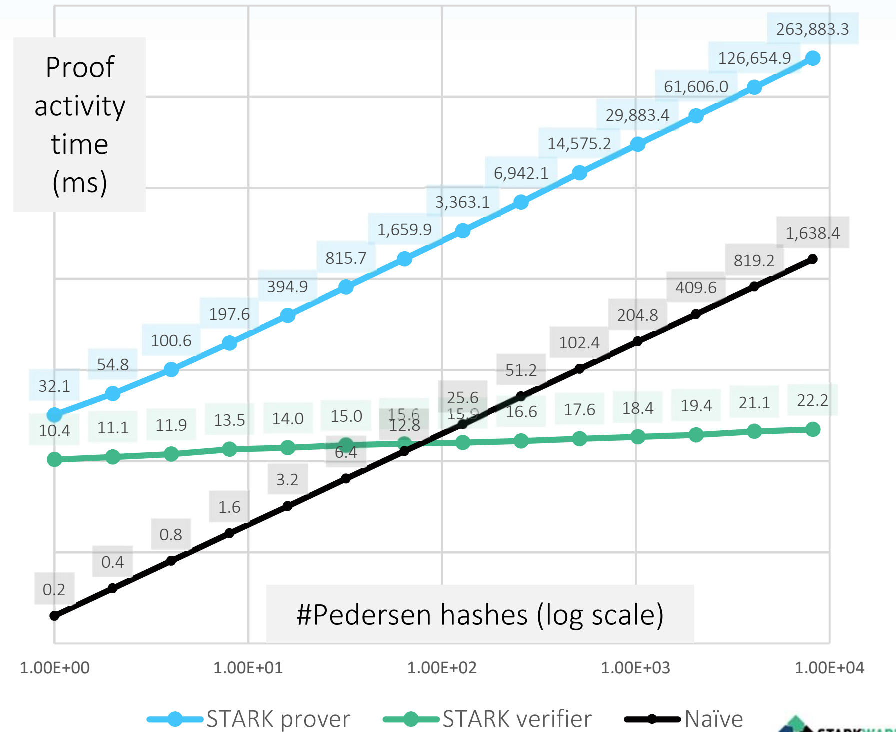
Scalability - Stark vs naive verification



Scalability - Stark vs naive verification

T440 laptop
i7-8550U CPU
@ 1.80GHz
Single thread
32 GB DDR4 RAM



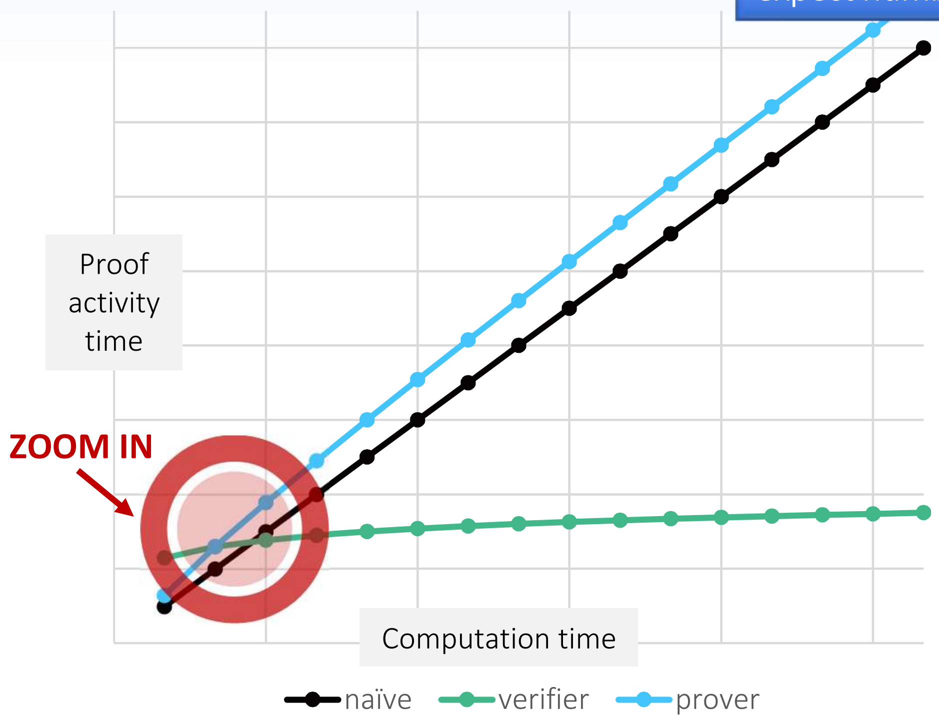


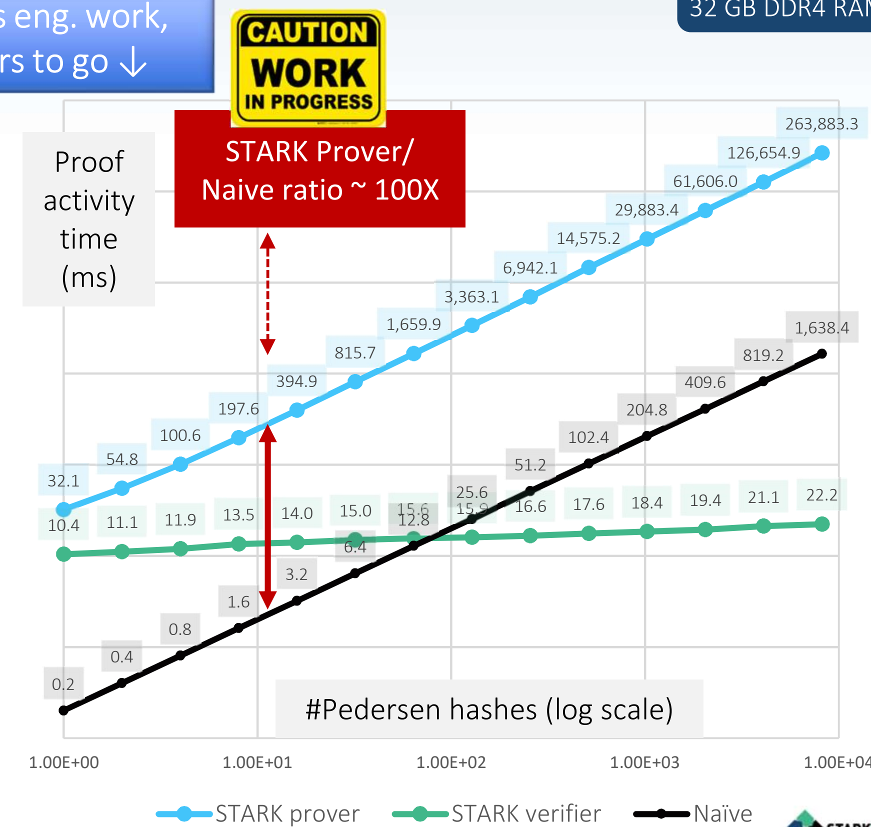
STARKWARE

Scalability - Stark vs naive verification

T440 laptop
i7-8550U CPU
@ 1.80GHz
Single thread
32 GB DDR4 RAM

after 6 months eng. work, expect numbers to go ↓





Virtues of Transparency (no trusted setup)

- Eliminate single point of failure (trusted setup)
- Facilitates continuous deployment of minor upgrades
- Reduces trust assumption: nothing up your sleeve, no need to trust setup ceremony
- Additional virtues of reliance on symmetric cryptography
  - Post-quantum security
  - Faster proving and verification time
  - Simpler, more reliable trust assumptions (CRH/Fiat-Shamir)
  - Reliance on "old" peer-reviewed principles (PCP Theorem, interactive knowledge extractors, …)

STARKWARE

STARK for Scalability

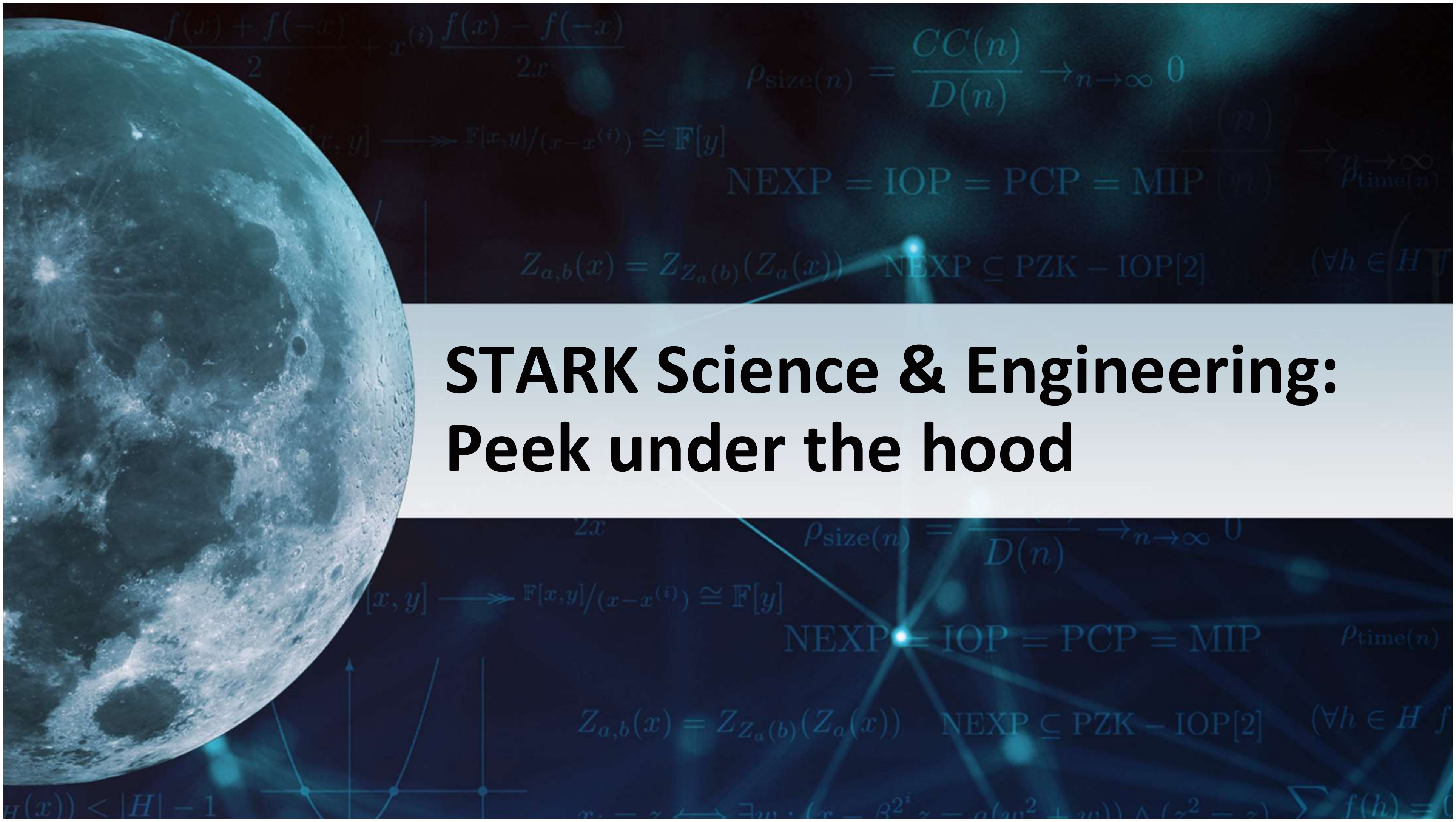
Trust Assumptions	Crypto Assumptions	Quantum Safety
ZK-STARK	ZK-STARK	ZK-STARK
ZK-SNARK	ZK-SNARK	ZK-SNARK
Bulletproof	Bulletproof	Bulletproof

ZK-STARK **ZK-SNARK** **Bulletproof**

STARKWARE

STARK for Scalability: Space

Runtime Comparison	ZK-STARK	ZK-SNARK	Bulletproof
1 Tx	~~500kb~~ ~~80kb~~ 45 kb (yet to identify lower bound)	Tx: 200 byte Key: 50 MB	1.5 kb
10K Tx	~~190kb~~ 135 k (yet to identify lower bound)	Tx: 200 byte Key: 500 GB	2.5 kb



STARK Science & Engineering: Peek under the hood

How to build efficient STARKs?

Convert computation to Algebraic Intermediate Representation (AIR)

- Generalization of R1CS constraints (used by SNARKs)
- State of computation is sequence of field elements
- Program expressed as set of polynomial relations over consecutive states
- Then apply more crypto+algebra …

Long term

tool-chain converting programs to AIRs

Mid term

domain specific languages for composing crypto-primitives

Short term

hand-optimize AIRs for specific computations

STARKWARE

Example: Pedersen Merkle path

Statement proved → Useful for.. → StarkWare's first major milestone → Let's examine the numbers

Statement proved:

I know a leaf in Merkle-tree of depth *d* with Merkle root *r*

Useful for..

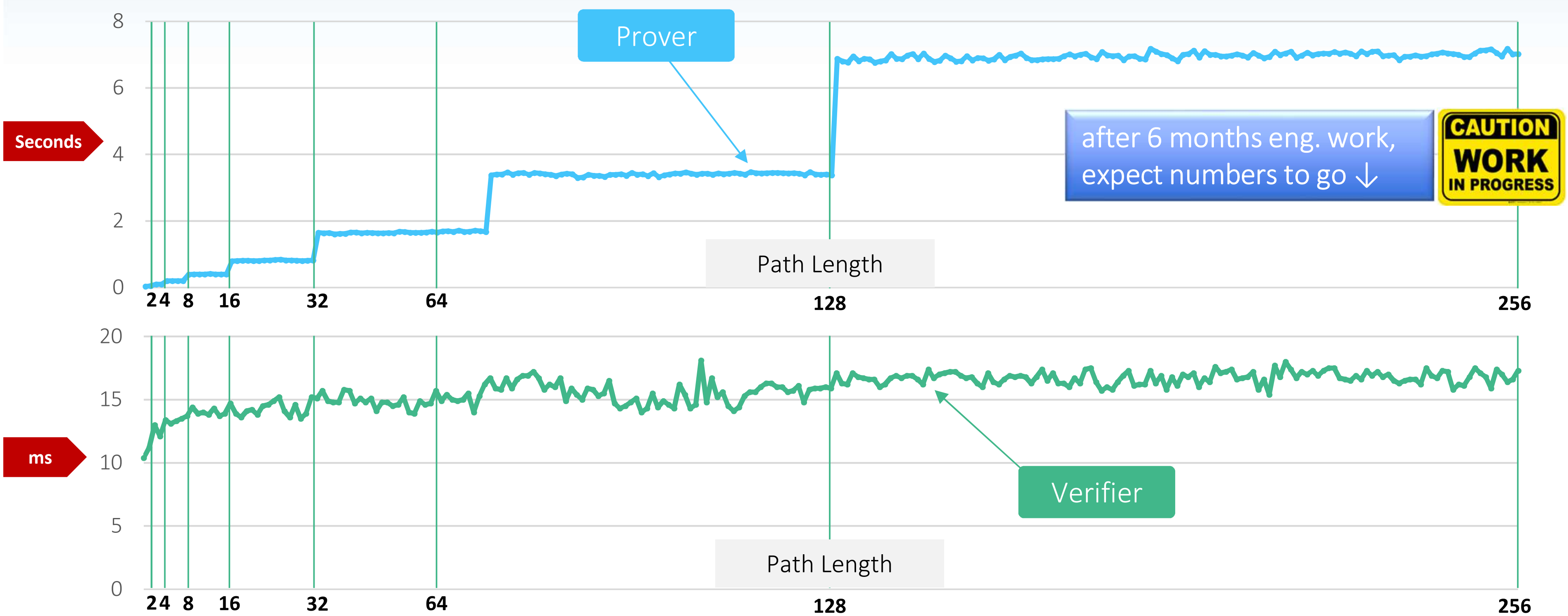
- Shielded Txs (major component in Zcash Sapling circuit)
- Verifiable Delay Functions (VDFs)
- Scalability solutions (more on this later)

StarkWare's first major milestone:

implementing STARK for this

STARK measurements - Pedersen Merkle path

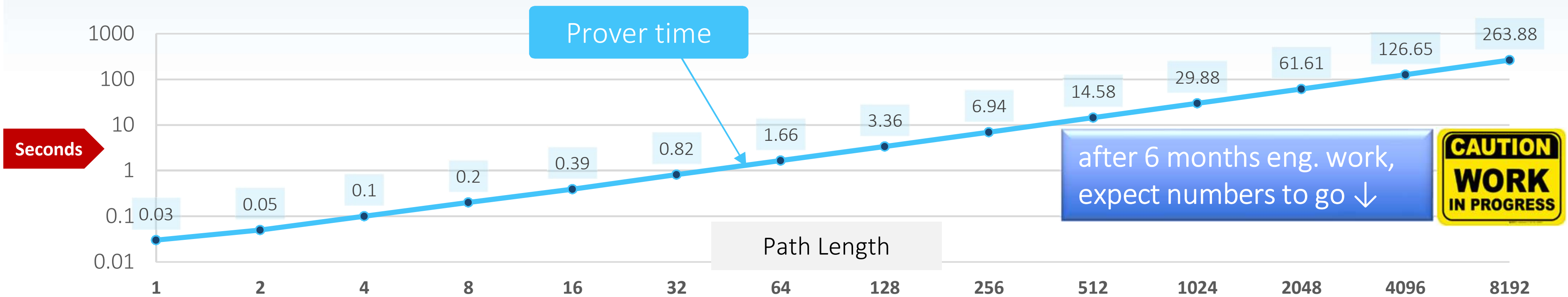
T440 laptop
i7-8550U CPU
@ 1.80GHz
Single thread
32 GB DDR4 RAM

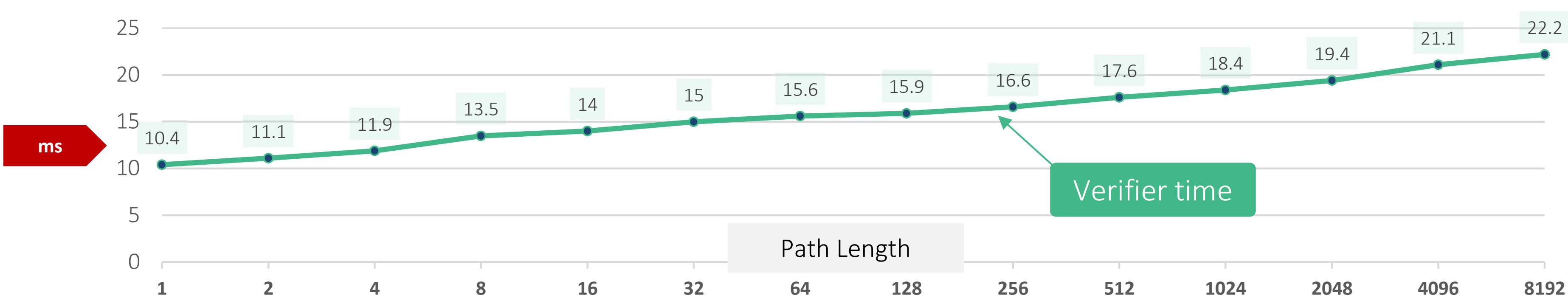


STARKWARE

STARK measurements - Pedersen Merkle path

T440 laptop
i7-8550U CPU
@ 1.80GHz
Single thread
32 GB DDR4 RAM

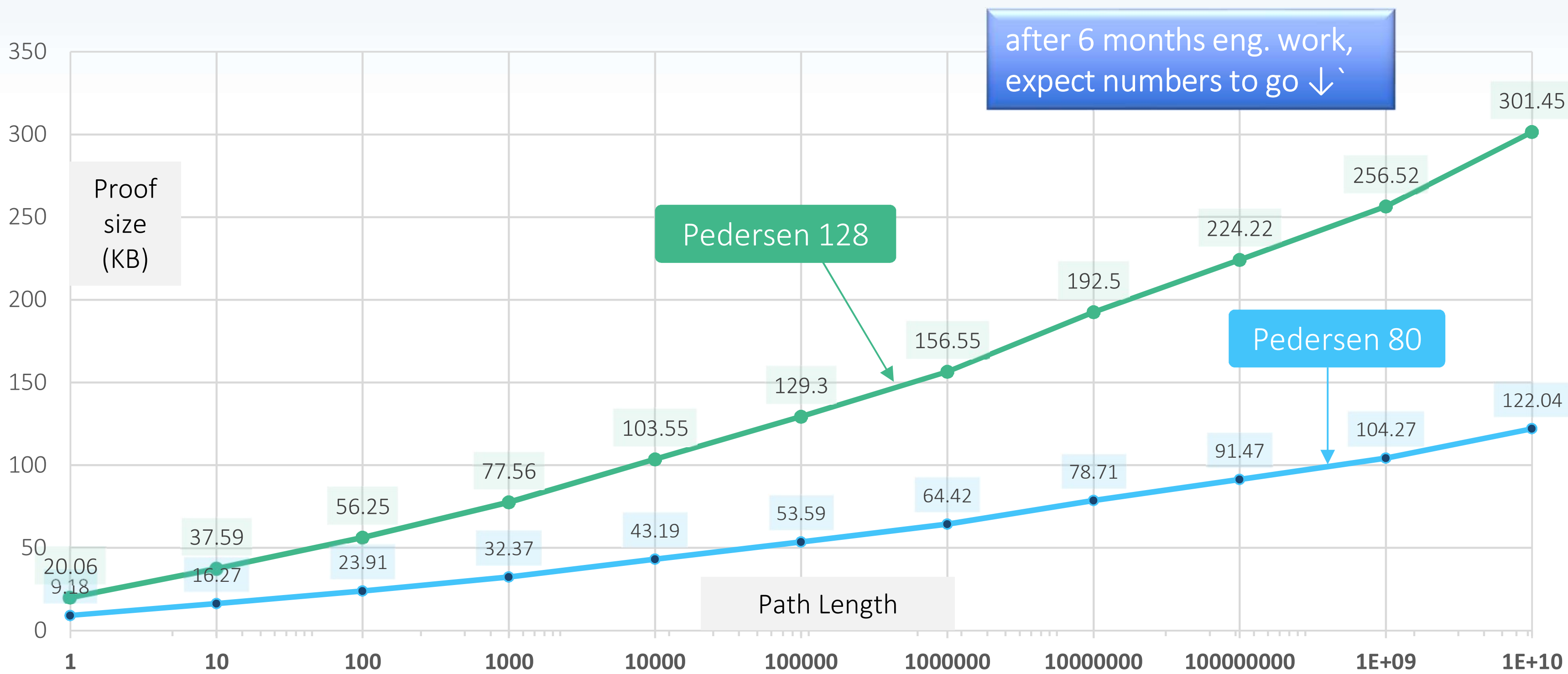




STARKWARE

STARK measurements - Pedersen Merkle path

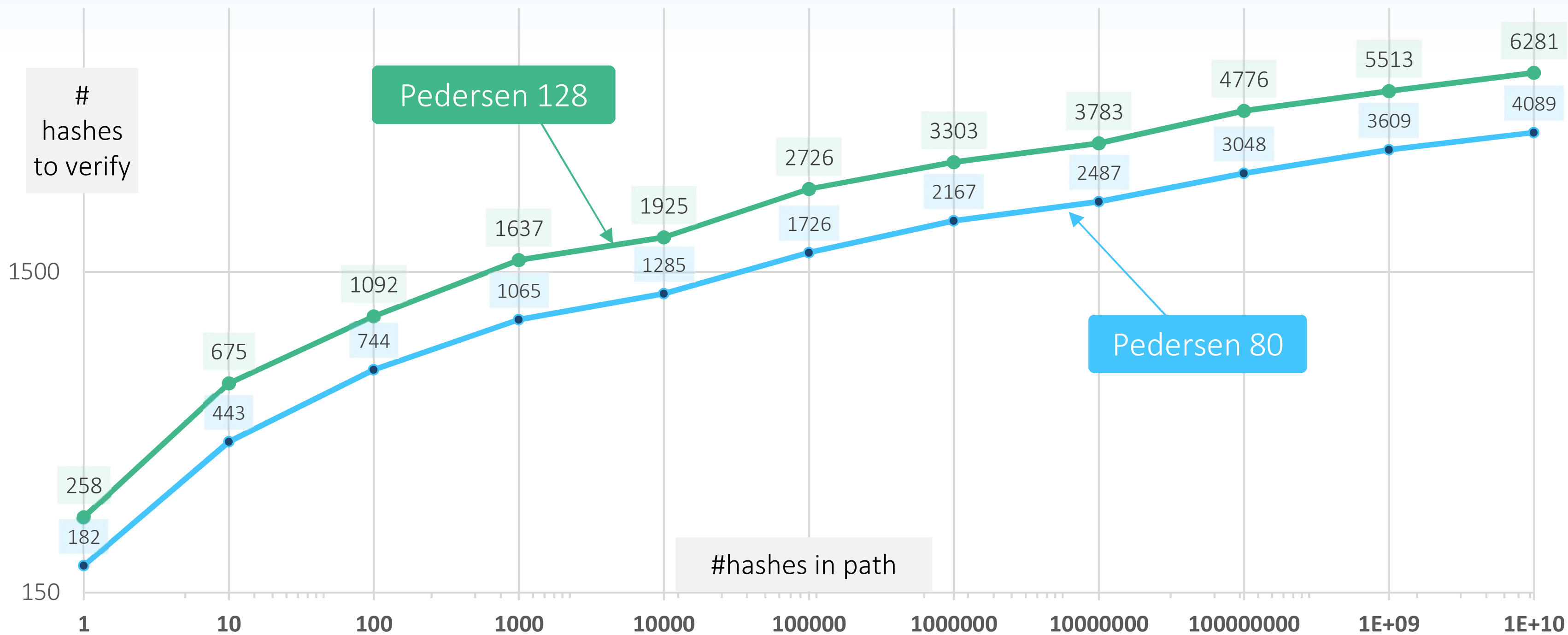
CAUTION WORK IN PROGRESS



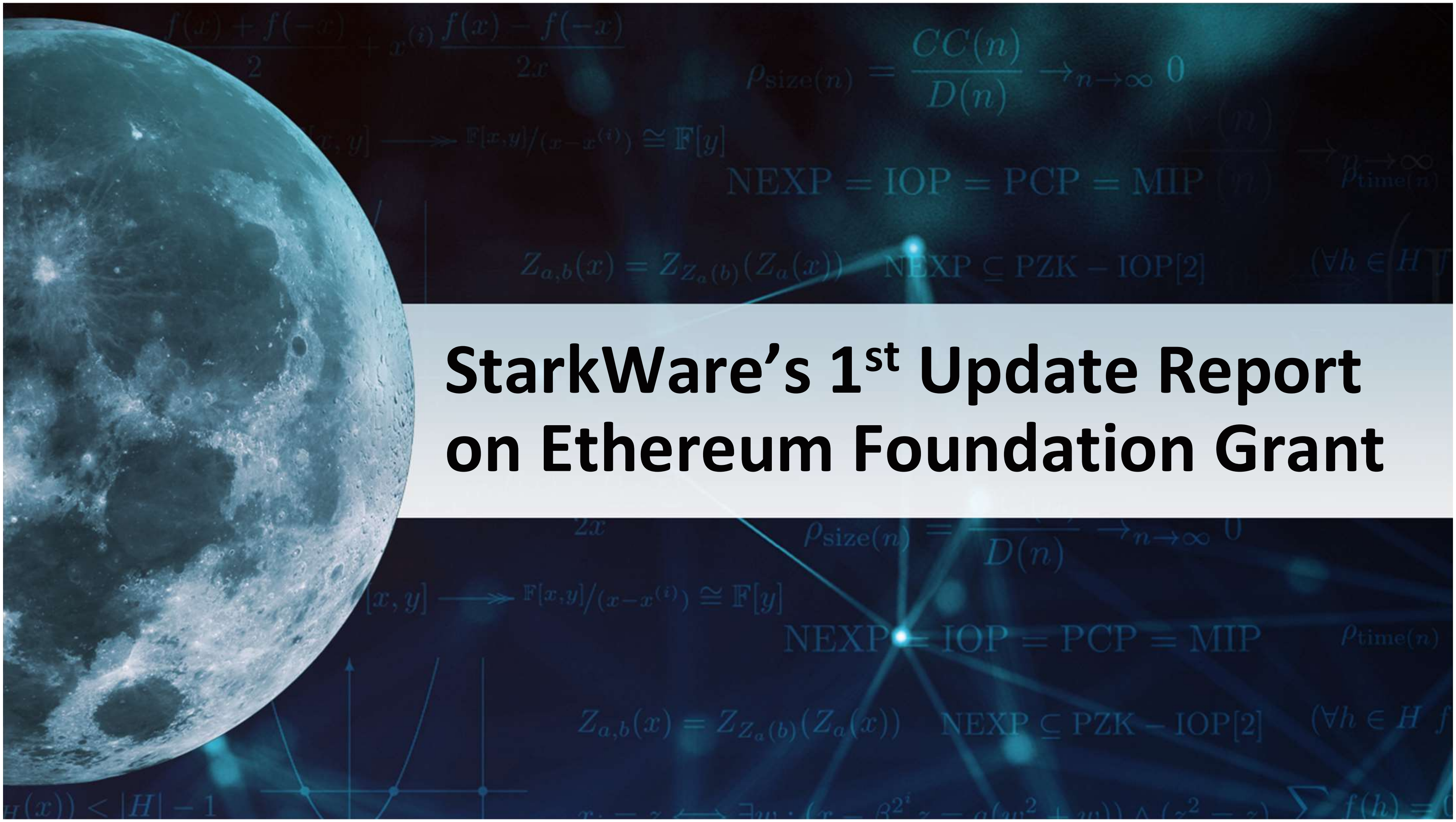
STARK verifier complexity

after 6 months eng. work, expect numbers to go ↓

CAUTION WORK IN PROGRESS



STARKWARE



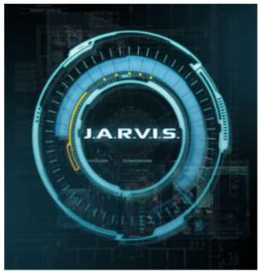
StarkWare's 1st Update Report on Ethereum Foundation Grant

1st T4T: Ethereum Foundation grant update

2-year performance and milestone-based grant

Requested Milestone: STARK proofs at rate of 100 hash/sec on quad-core

Latest: 100 Pedersens/3 sec on single thread of this laptop (1.8Ghz)



STARK-friendly hash functions

Defined complexity parameters of STARK friendly primitives

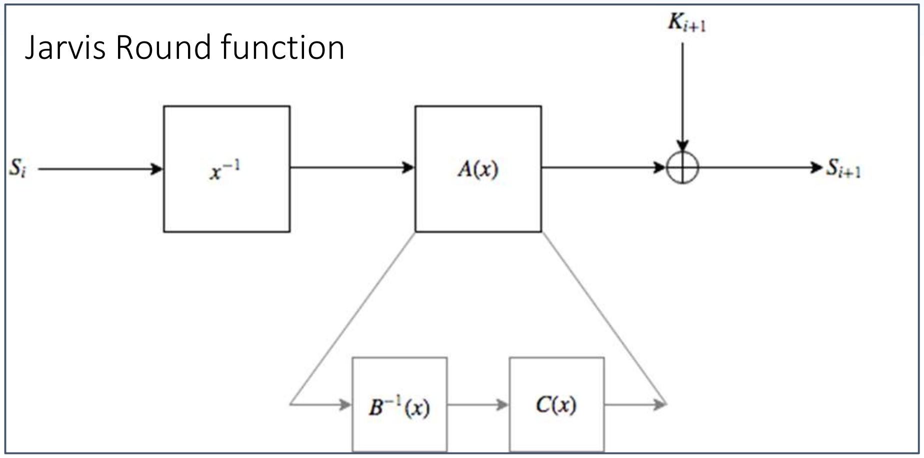
Designed by *Dr. Tomer Ashur & Siemen Dhooghe*, Scientific supervision by *Prof. Vincent Rijmen* (co-inventor of Rijndael/AES cypher standard), see [Tomer's blog]

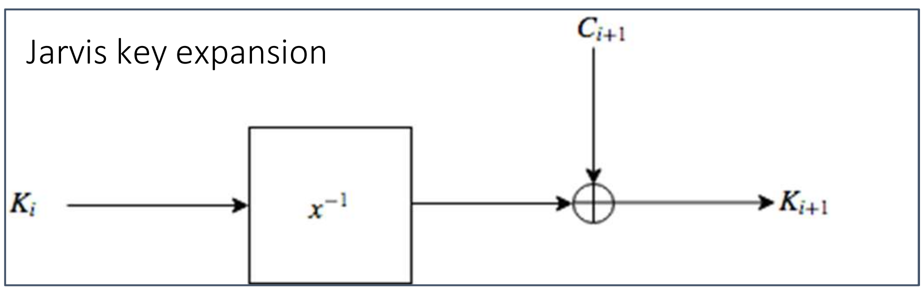
- Rijndael/AES-based constructions (over binary fields)
- Jarvis - STARK-friendly cypher candidate
- Friday - STARK-friendly hash candidate, ~25X STARK-friendlier than Pedersen
- Next: expert+community review, including cypher-breaking competition

STARKWARE

1st T4T: Ethereum Foundation grant update

Images from Dr. Tomer Ashur's blog:
www.esat.kuleuven.be/cosic/jarvis-and-friday-stark-friendly-cryptographic-primitives/



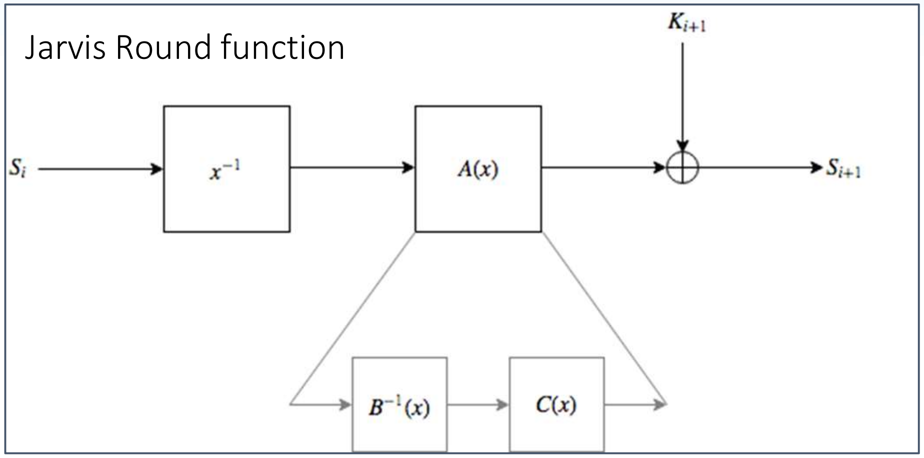


- Jarvis is a cypher, Friday is its hash (Miyaguchi-Perneel)
- Jarvis defined over GF(2^{2n}) for n-bit security
- # rounds TBD, between 6 to 10
- B(X), C(X): quartic linearized permutation polynomials
- Security relies on Rijndael-style analysis
- Jarvis is not prime-field friendly (not SNARK/BP friendly)

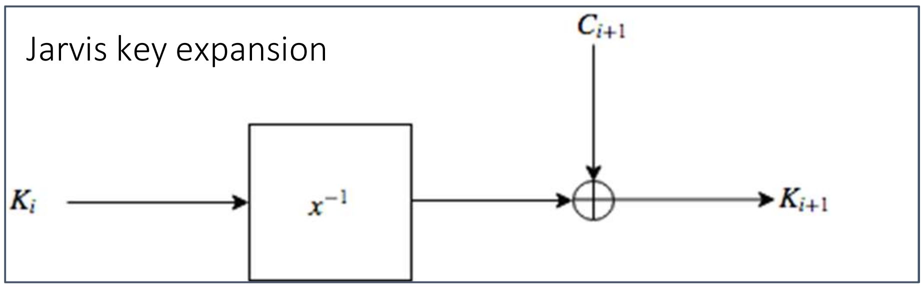
1st T4T: Ethereum Foundation grant update

[Images from Dr. Tomer Ashur's blog](www.esat.kuleuven.be/cosic/jarvis-and-friday-stark-friendly-cryptographic-primitives/):
www.esat.kuleuven.be/cosic/jarvis-and-friday-stark-friendly-cryptographic-primitives/

Jarvis Round function



Jarvis key expansion



To support STARK, Jarvis, Friday, need

- EVM opcodes for binary field +, *, /
- EVM opcodes for blake, Friday
- We'll submit an EIP request

How to build efficient STARKs?

Convert computation to Algebraic Intermediate Representation (AIR)

Generalization of R1CS constraints (used by SNARKs)

State of computation is sequence of field elements

Algebraic Execution Trace (AET) captures computation:

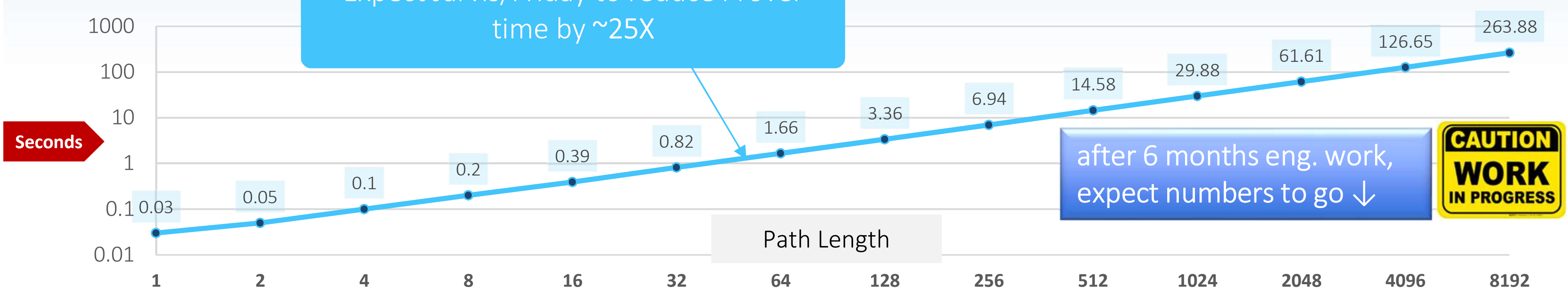
	security	width	cycles	degree	w*c*d
SHA2	128	56	3762	11	2,317,392
Rijdael 160	80	68	58	8	31,552
Pedersen	128	4	256	2	2,048
MimC	128	1	70-80	3	210 - 240
Friday	128	5	6-10	2	60 - 100

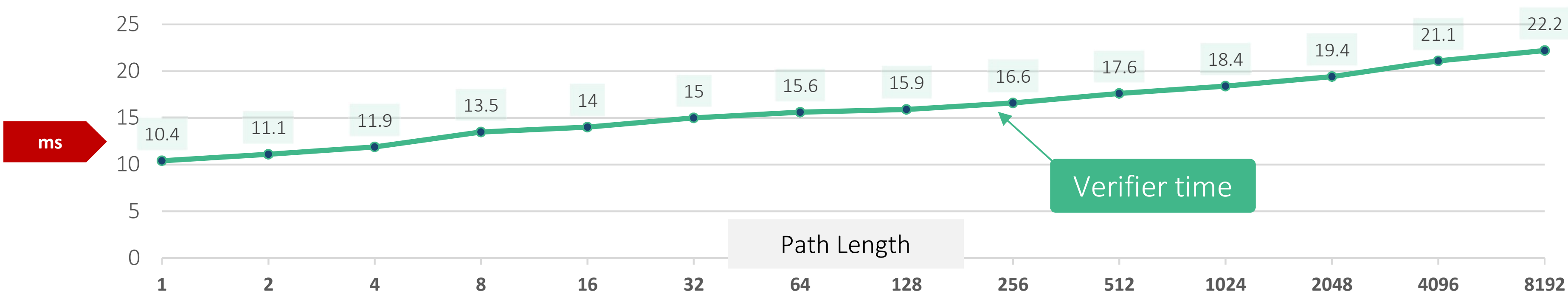
STARKWARE

STARK measurements - Pedersen Merkle path

T440 laptop
i7-8550U CPU
@ 1.80GHz
Single thread
32 GB DDR4 RAM

Expect Jarvis/Friday to reduce Prover time by ~25X





STARKWARE

STARK proof length estimates, repeated hash

CAUTION
WORK
IN PROGRESS

after 6 months eng. work, expect numbers to go ↓

STARKWARE


STARKWARE
Demo


STARKWARE
One more thing…


STARKWARE
Questions?

STARKWARE

STARK Use Cases

Avihu Levy, Head of Product | October 2018

Potential Uses

Scalability → Privacy → Bootstrap & Sync → STARK-based primitives → …

Scalability

- Infrastructure level (layer 1 / layer 2)
- Application level (STARK-powered dapp)

STARK-based primitives

- VDF
- Signature aggregation

Scalability

Complex computations

Batches of simple computations

Verifying 10K Txs costs only 3x of verifying single Tx!

Storage

What’s Done On-Chain Vs. Off-Chain?

Verifier

On Chain

Verifier

Prover

Off Chain

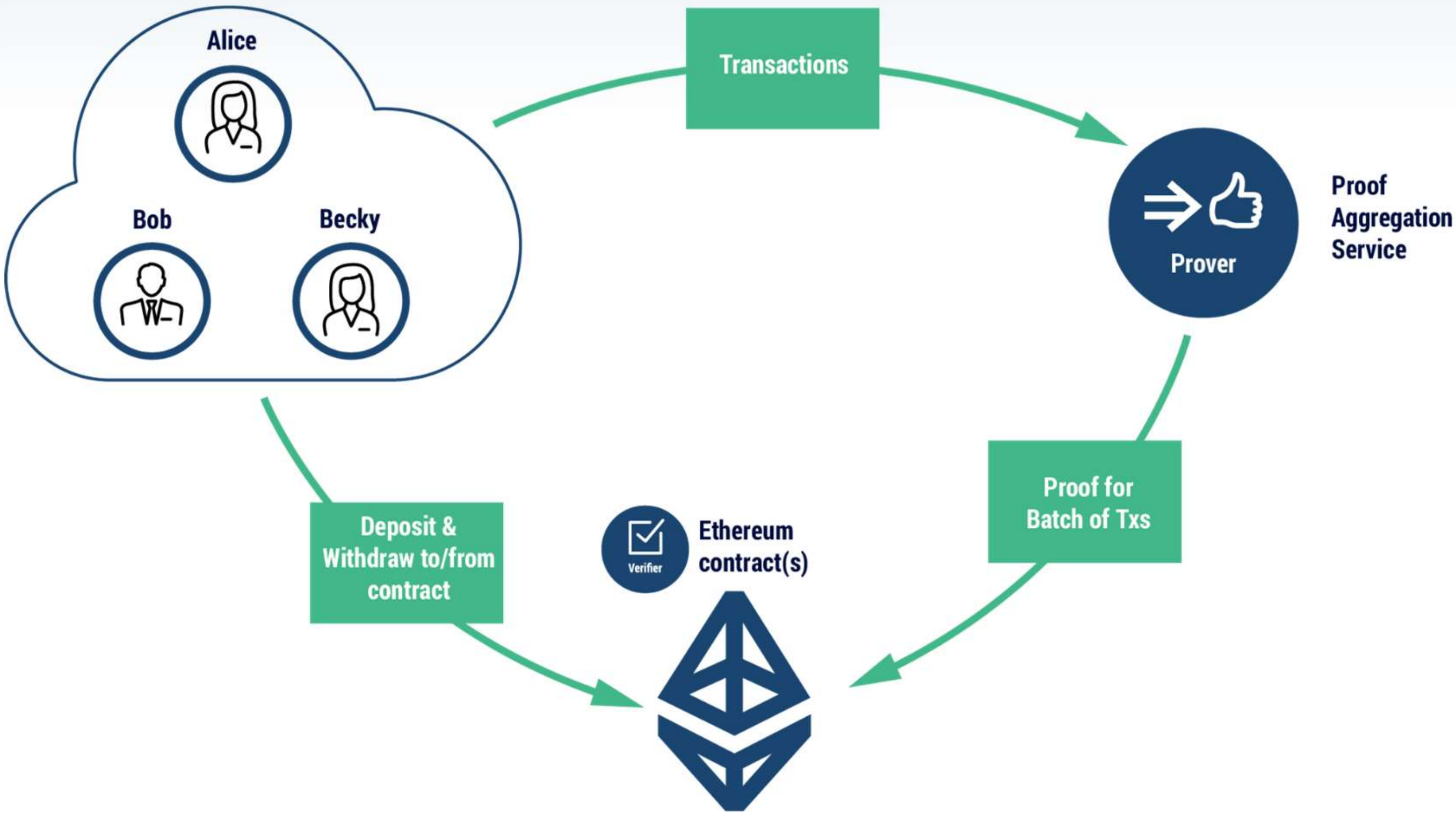
Native

Computation

Prover

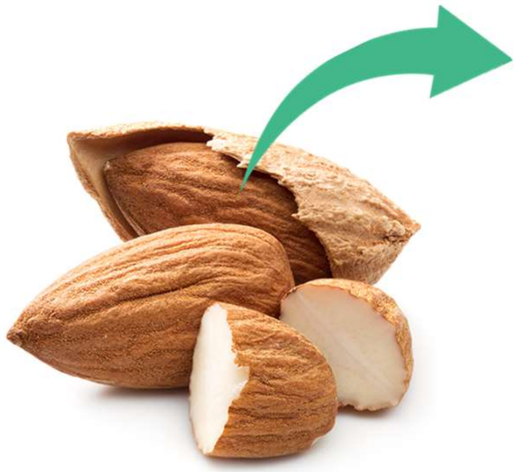
STARKWARE

Scalability by batching



STARKWARE

Payment Txs: In a Nutshell



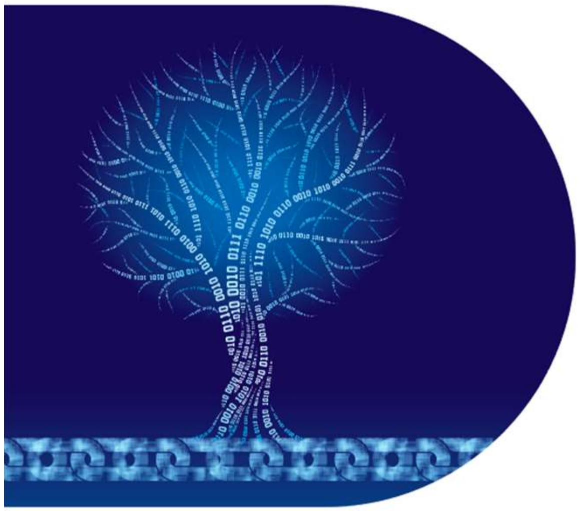
Increase the network throughput (e.g. of Ethereum)

Reduce Tx fees
(for native Ether and
ERC-20 tokens)

Significant saving
(> 10X)

**Users register to the contract,
and only then can interact with one another**

Payment Txs: How Is This Achieved?



No need to send and verify signatures

Covered by proof

Less transmission, less computation

State storage taken off chain, state root stored on-chain

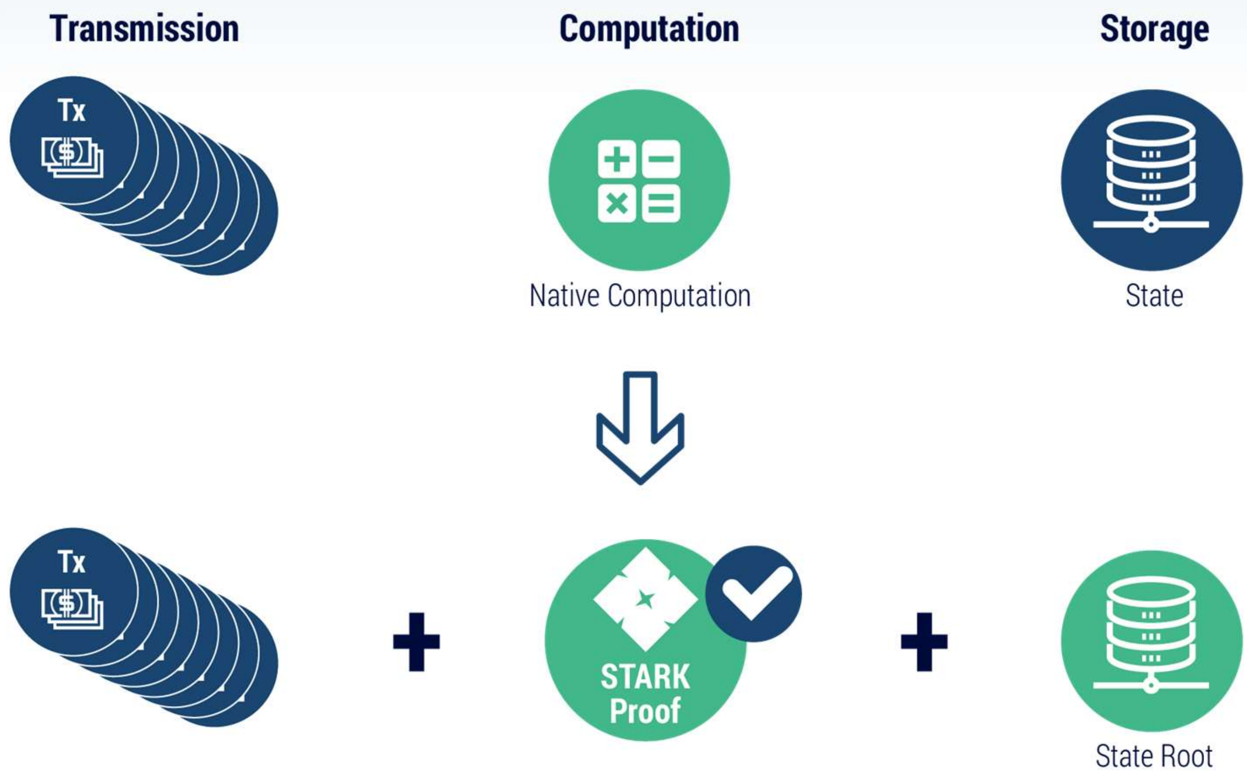
Proof proves state changes

Centralization trade-off (unless done by miners)

Increasing network throughput by 1 OOM

STARKWARE

Scalability



DEX: In a Nutshell

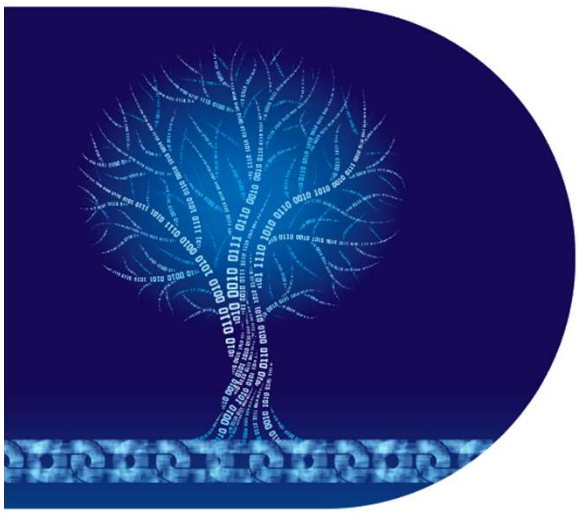
Provide users with “best of all worlds”:

- Users keep custody of their funds
- Low latency
- At a negligible blockchain cost
- Breaking the gas-induced ceiling on volume/liquidity

Later add (cheap) transfers and shielded Txs through DEX

STARKWARE

DEX: How Is This Achieved?



No need to send and verify maker/taker signatures

Covered by proof

Less transmission, less computation

Trader balances taken off chain, state root stored on-chain

Proof proves balance changes

Centralization not an issue in this case

lowering blockchain costs* by 1-2 OOM

*Blockchain costs are currently the dominant DEX costs

Overview

- What's a STARK?
- STARK comparison to
  - SNARKs (Zcash)
  - Recursive SNARKs (Coda)
  - Bulletproofs (Monero)
- Benefits of STARK for scalability
- Potential use-cases


STARKWARE

Bootstrap & Sync

Problem:
new clients bootstrapping process is heavy, as well as keeping your client sync

$\Rightarrow$

Solution:
Download a state and a proof for the validity of the state, rather than download & recompute the whole history

Bootstrap & Sync

Block headers for light clients

E.g. (Ethereum):
~500 bytes header,
~3MB headers/day

Instead, a stark proof for block header having X work behind it since genesis/ last known checkpoint

Proof size ~100KB with 20 ms verification time

Full Clients

Prove that the state computed based on previous block state/ genesis block is valid

VDF

Problem:
randomness from block data is hard, if the block creator knows the output of the random function

⇒

Solution:
a function that take time to compute, so when fixing the input the output is unknown

We do want a fast verification of the output

STARKWARE

VDF