From NAND to Verifiable TETRIS

INGONYAMA

Omer Shlomovits , ZKProof 5 Workshop

Scope

Scope

Algorithms	Software	Hardware

Scope

Algorithms	Software	Hardware
Groth16		

Scope

Algorithms	Software	Hardware
Groth16	C++ implementation • We can run G1,G2 in parallel	

Scope

Algorithms	Software	Hardware
Groth16	C++ implementation • We can run G1,G2 in parallel	Clock runs at frequency f. • Here is a circuit that runs G1 in minimal clock cycles • Here is a machine that runs G2 under area A[mm^2] • Here is a machine that runs Groth16 taking <5[Watts]

Scope

Algorithms	Software	Hardware
Groth16	C++ implementation • We can run G1,G2 in parallel	Clock runs at frequency f. • Here is a circuit that runs G1 in minimal clock cycles • Here is a machine that runs G2 under area A[mm^2] • Here is a machine that runs Groth16 taking <5[Watts]

- For HW to make sense, we must focus on real-world systems

Scope

Algorithms	Software	Hardware
Groth16	C++ implementation • We can run G1,G2 in parallel	Clock runs at frequency f. • Here is a circuit that runs G1 in minimal clock cycles • Here is a machine that runs G2 under area A[mm^2] • Here is a machine that runs Groth16 taking <5[Watts]

- For HW to make sense, we must focus on real-world systems
- Bias towards SNARKs, STARKs

Scope

Algorithms	Software	Hardware
Groth16	C++ implementation • We can run G1,G2 in parallel	Clock runs at frequency f. • Here is a circuit that runs G1 in minimal clock cycles • Here is a machine that runs G2 under area A[mm^2] • Here is a machine that runs Groth16 taking <5[Watts]

- For HW to make sense, we must focus on real-world systems
- Bias towards SNARKs, STARKs
- Proving time is the main bottleneck

ZK Hardware Landscape

Much action, much stealth

PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture

Ye Zhang[1,5] Shuo Wang[1] Xian Zhang[3] Jiangbin Dong[4,7] Xingzhong Mao[7] Fan Long[6]
Cong Wang[8] Dong Zhou[2] Mingyu Gao[2,7*] Guangyu Sun[1*]

PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture

Ye Zhang[1,5] Shuo Wang[1] Xian Zhang[3] Jiangbin Dong[4,7] Xingzhong Mao[7] Fan Long[6]
Cong Wang[8] Dong Zhou[2] Mingyu Gao[2,7*] Guangyu Sun[1*]

PipeMSM: Hardware Acceleration for Multi-Scalar Multiplication

Charles. F. Xavier
charlie@ingonyama.com

PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture

Ye Zhang[1,5] Shuo Wang[1] Xian Zhang[3] Jiangbin Dong[4,7] Xingzhong Mao[7] Fan Long[6]
Cong Wang[8] Dong Zhou[2] Mingyu Gao[2,7*] Guangyu Sun[1*]

PipeMSM: Hardware Acceleration for Multi-Scalar Multiplication

Charles. F. Xavier
charlie@ingonyama.com

FPGA Acceleration of Multi-Scalar Multiplication: CycloneMSM

Kaveh Aasaraai, Don Beaver, Emanuele Cesena, Rahul Maganti,
Nicolas Stalder and Javier Varela

PipeZK: Accelerating Zero-Knowledge Proof with a Pipelined Architecture

Ye Zhang[1,5] Shuo Wang[1] Xian Zhang[3] Jiangbin Dong[4,7] Xingzhong Mao[7] Fan Long[6]
Cong Wang[8] Dong Zhou[2] Mingyu Gao[2,7*] Guangyu Sun[1*]

PipeMSM: Hardware Acceleration for Multi-Scalar Multiplication

Charles. F. Xavier
charlie@ingonyama.com

FPGA Acceleration of Multi-Scalar Multiplication: CycloneMSM

Kaveh Aasaraai, Don Beaver, Emanuele Cesena, Rahul Maganti,
Nicolas Stalder and Javier Varela

cuZK: Accelerating Zero-Knowledge Proof with A Faster Parallel Multi-Scalar Multiplication Algorithm on GPUs

Tao Lu, Chengkun Wei, Ruijing Yu, Yi Chen, Li Wang, Chaochao Chen,
Zeke Wang, and Wenzhi Chen

ZPRIZE

Prizes

* General interest prize category for public goods which benefit multiple protocols/proof systems

OPEN DIVISION	OPEN DIVISION	OPEN DIVISION
Accelerating MSM Operations on GPU/FPGA	Accelerating NTT Operations on an FPGA	Plonk-DIZK GPU Acceleration

PRIZE AMOUNT **$1,250,000 USD**

Why Hardware Acceleration

A brief detour into gaming

Quake, 1997

Call of Duty, 2022

The difference - GPU

Multiplayer games

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
    - Ownership
    - Expensive
    - Scalability
    - Privacy

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
    - Ownership
    - Expensive
    - Scalability
    - Privacy
- **P2P** - players can cheat reporting on their state

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
    - Ownership
    - Expensive
    - Scalability
    - Privacy
- **P2P** - players can cheat reporting on their state

ZK fixes this 💪

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
  - Ownership
  - Expensive
  - Scalability
  - Privacy
- **P2P** - players can cheat reporting on their state

ZK fixes this 💪

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
    - Ownership
    - Expensive
    - Scalability
    - Privacy
- **P2P** - players can cheat reporting on their state

ZK fixes this 💪

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
  - Ownership
  - Expensive
  - Scalability
  - Privacy
- **P2P** - players can cheat reporting on their state

ZK fixes this 💪

Problem: we need to run many ZKPs per sec for state changes

Multiplayer games

- Two ways to run a multiplayer game: (1) client-server, (2) P2P
- **Client-server (common)** - centralized (trusted) authoritative server
  - Ownership
  - Expensive
  - Scalability
  - Privacy
- **P2P** - players can cheat reporting on their state

ZK fixes this 💪

Happy to Discuss!

Problem: we need to run many ZKPs per sec for state changes

Are we there yet?

Are we there yet?

- **CryptoPunk ownership:** 25[s] proof generation time [1]
- **RISC-V ZKVM:** 30k - 1M instructions per second (70's computer speed)[2]
- **ZK MLaaS of ImageNet scale inference:** 2457[s] proving time at <80% accuracy [3]
- **Scroll ZKEVM:** 2535[s] for EVM circuit prover [4]
- **zkBridge** costs 8100 USD per month to operate [5]

Are we there yet?

- **CryptoPunk ownership:** 25[s] proof generation time [1]
- **RISC-V ZKVM:** 30k - 1M instructions per second (70's computer speed)[2]
- **ZK MLaaS of ImageNet scale inference:** 2457[s] proving time at <80% accuracy [3]
- **Scroll ZKEVM:** 2535[s] for EVM circuit prover [4]
- **zkBridge** costs 8100 USD per month to operate [5]

Answer: We are not far!

ZPU

Zero-Knowledge Processing Unit

ZPU

Zero-Knowledge Processing Unit

1. What goes into a ZPU? (hint: standardization can help)
2. What can be done until we have ZPUs?

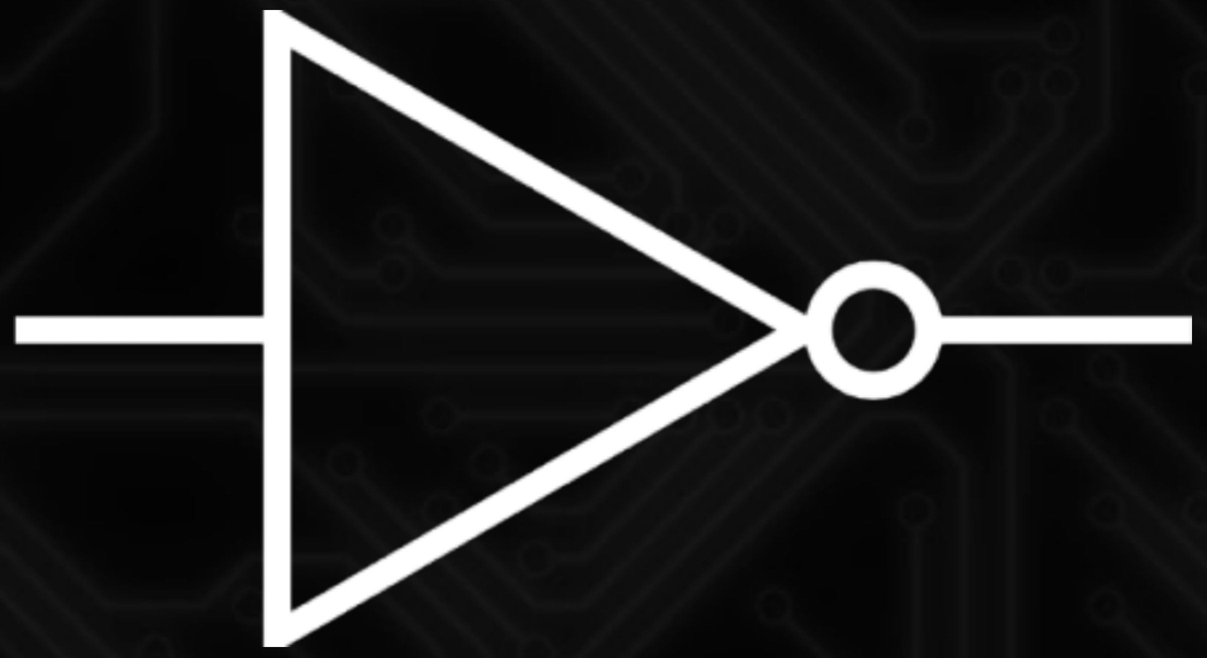
ZPU

How to identify one when you see one?

ZPU

How to identify one when you see one?

- AES-NI but for ZK
  - Fixed standardized algorithm vs. moving, evolving target
- A dedicated CPU
  - Good for integer and bitwise arithmetic, we work with finite fields



ZPU

How to identify one when you see one?

- Chip for modular arithmetic {add, mul, inv}
  - What field? Can we add specific primitives?

ZPU

How to identify one when you see one?

- Chip for modular arithmetic {add, mul, inv}
  - What field? Can we add specific primitives?
- Polynomial arithmetic over finite field(s) (FFT acceleration)
- Elliptic curve(s) operations acceleration (MSM acceleration)
- Vectorized arithmetic (Hash function(s) acceleration)

ZPU

How to identify one when you see one?

- Chip for modular arithmetic {add, mul, inv}
  - What field? Can we add specific primitives?
- Polynomial arithmetic over finite field(s) (FFT acceleration)
- Elliptic curve(s) operations acceleration (MSM acceleration)
- Vectorized arithmetic (Hash function(s) acceleration)

Small area
Less performance
Less Power

Large area
More performance
More Power

Prioritized list for standardization for ZKProof6

Opinionated list!

- Finite Field/s arithmetic:
  - (e.g. 64bit prime vs. 256bit prime)
  - Elliptic Curve
- NTT-less (STARK vs. SNARK)
  - Standardize Protocol/s
- Vector operations:
  - ZK-friendly (arithmetic) hash function/s
  - Arithmetization (Plonk style vs. R1CS style)
- Range of circuit size

Prioritized list for standardization for ZKProof6

Opinionated list!

- Finite Field/s arithmetic:
  - (e.g. 64bit prime vs. 256bit prime)
  - Elliptic Curve
- NTT-less (STARK vs. SNARK)
  - Standardize Protocol/s
- Vector operations:
  - ZK-friendly (arithmetic) hash function/s
  - Arithmetization (Plonk style vs. R1CS style)
- Range of circuit size

Happy to Discuss!

What can be done today?

What can be done today?

- Do Science
- Build solutions on existing Hardware

Research Questions

A selection

Research Questions

A selection

- What is the optimal HW-Friendly ZK protocol?

Research Questions

A selection

- What is the optimal HW-Friendly ZK protocol?
- Can we design a hash function that is both ZK-friendly and HW-friendly?

Research Questions

A selection

- What is the optimal HW-Friendly ZK protocol?
- Can we design a hash function that is both ZK-friendly and HW-friendly?
- Imagine a world where prover computing is no longer a bottleneck… now what?

Research Questions

A selection

- More HW-Friendly ZK protocols
- ZK-friendly & HW-friendly hash functions
- Imagine a world where prover computing is no longer a bottleneck... now what?
- More Memory awareness/management in SNARKs/ STARKSs

Research Questions

A selection

- More HW-Friendly ZK protocols
- ZK-friendly & HW-friendly hash functions
- Imagine a world where prover computing is no longer a bottleneck... now what?
- More Memory awareness/management in SNARKs/ STARKSs

Happy to Discuss!

Build

- Available Hardware
- What to build?
- Examples

Available Hardware

GPUs

FPGAs

Available Hardware

GPUs

- Cheap to buy

FPGAs

Available Hardware

GPUs

- Cheap to buy

FPGAs

- Cheap to operate

Available Hardware

GPUs

- Cheap to buy
- Mining GPUs can be repurposed to run ZK

FPGAs

- Cheap to operate

Available Hardware

GPUs

- Cheap to buy
- Mining GPUs can be repurposed to run ZK

FPGAs

- Cheap to operate
- FPGA data centers can be utilized

Available Hardware

GPUs

- Cheap to buy
- Mining GPUs can be repurposed to run ZK
- Available on-demand in the cloud

FPGAs

- Cheap to operate
- FPGA data centers can be utilized
- Available on-demand in the cloud

Name	FPGAs	vCPUs	Instance Memory (GiB)	SSD Storage (GB)	Enhanced Networking	EBS Optimized	On-Demand Price/hr*
f1.2xlarge	1	8	122	470	Yes	Yes	$1.65
f1.4xlarge	2	16	244	940	Yes	Yes	$3.30
f1.16xlarge	8	64	976	4 x 940	Yes	Yes	$13.20

Available Hardware

GPUs

- Cheap to buy
- Mining GPUs can be repurposed to run ZK
- Available on-demand in the cloud

- Empirically: Gaming GPUs perform better
  - Cons: Form factor, Energy

FPGAs

- Cheap to operate
- FPGA data centers can be utilized
- Available on-demand in the cloud

Name	FPGAs	vCPUs	Instance Memory (GiB)	SSD Storage (GB)	Enhanced Networking	EBS Optimized	On-Demand Price/hr*
f1.2xlarge	1	8	122	470	Yes	Yes	$1.65
f1.4xlarge	2	16	244	940	Yes	Yes	$3.30
f1.16xlarge	8	64	976	4 x 940	Yes	Yes	$13.20

Available Hardware

GPUs

- Cheap to buy
- Mining GPUs can be repurposed to run ZK
- Available on-demand in the cloud
- Empirically: Gaming GPUs perform better
  - Cons: Form factor, Energy
- Open-source libraries: e.g. sppark, ec-gpu

ec-gpu & ec-gpu-gen

crates.io v0.2.0 docs passing circleci passing rustc 1.54+

crates.io v0.4.0 docs passing circleci passing rustc 1.54+

CUDA/OpenCL code generator for finite-field arithmetic over prime fields and elliptic curve arithmetic constructed with Rust.

FPGAs

- Cheap to operate
- FPGA data centers can be utilized
- Available on-demand in the cloud

Name	FPGAs	vCPUs	Instance Memory (GiB)	SSD Storage (GB)	Enhanced Networking	EBS Optimized	On-Demand Price/hr*
f1.2xlarge	1	8	122	470	Yes	Yes	$1.65
f1.4xlarge	2	16	244	940	Yes	Yes	$3.30
f1.16xlarge	8	64	976	4 x 940	Yes	Yes	$13.20

- Open-source

cloud-ZK

A toolkit for developing ZKP acceleration in the cloud

Build

- **Infra & Foundations**
  - Prover-as-a-Service
    - Decentralized Prover-as-a-Service
  - ZK marketplace
  - App-specific, e.g. Filecoin miner, Danksharding Builder
- **Frameworks Integrations**, e.g. Arkworks, Circom

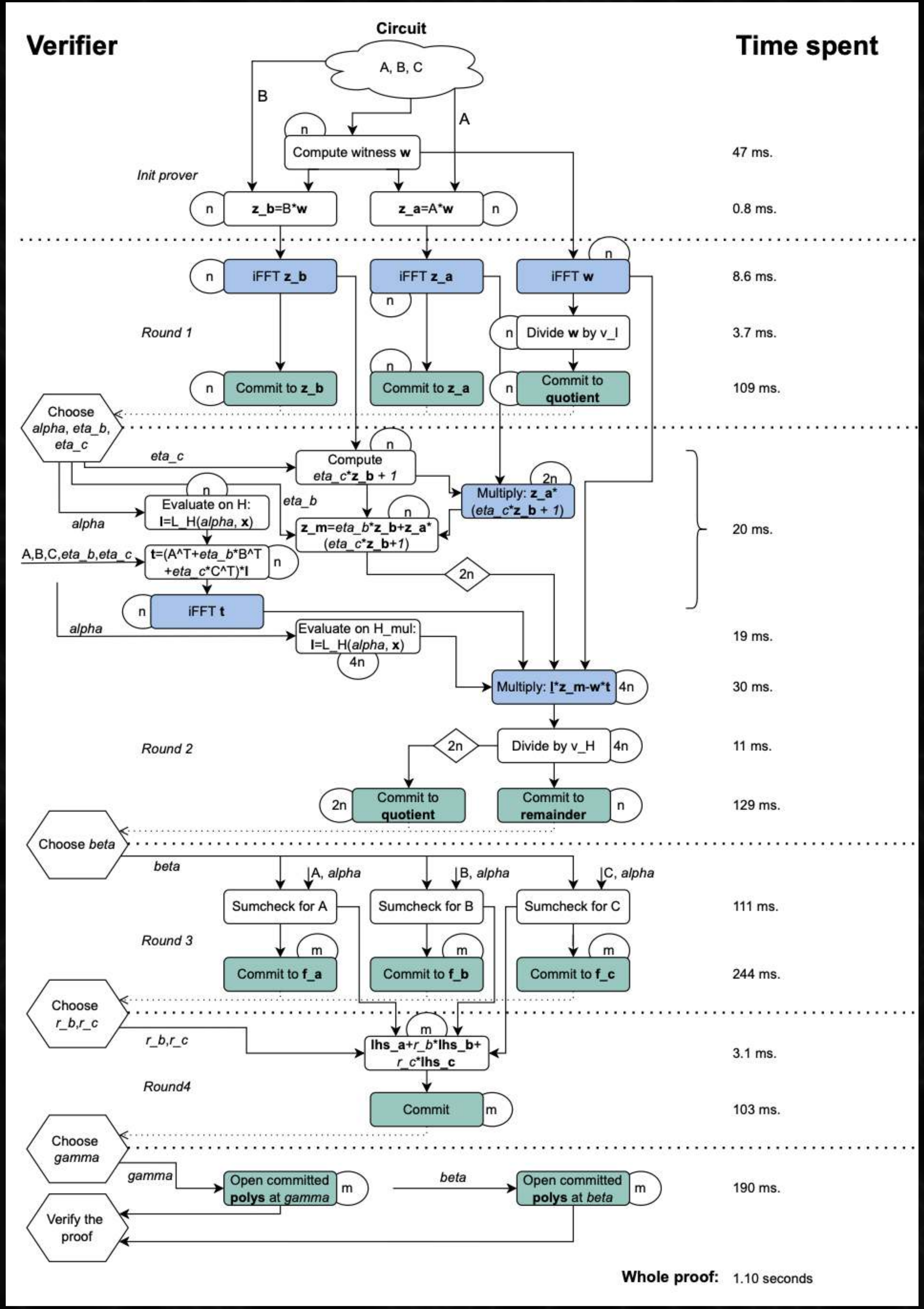
Example 1: Aleo miner

Example 1: Aleo miner

- GPU/FPGA/GPU+FPGA
  - Electricity cost is important factor

Example 1: Aleo miner

- GPU/FPGA/GPU+FPGA
  - Electricity cost is important factor
- Optimized Marlin Prover
  - BLS12-377
  - Mostly MSMs (70%) and FFTs (15%)



Example 1: Aleo miner

- GPU/FPGA/GPU+FPGA
  - Electricity cost is important factor
- Optimized Marlin Prover
  - BLS12-377
  - Mostly MSMs (70%) and FFTs (15%)

CPU: 0.9 [proofs/sec]

HW: 8 [proofs/sec]

Example 2: Rapid RapidSnark

Example 2: Rapid RapidSnark

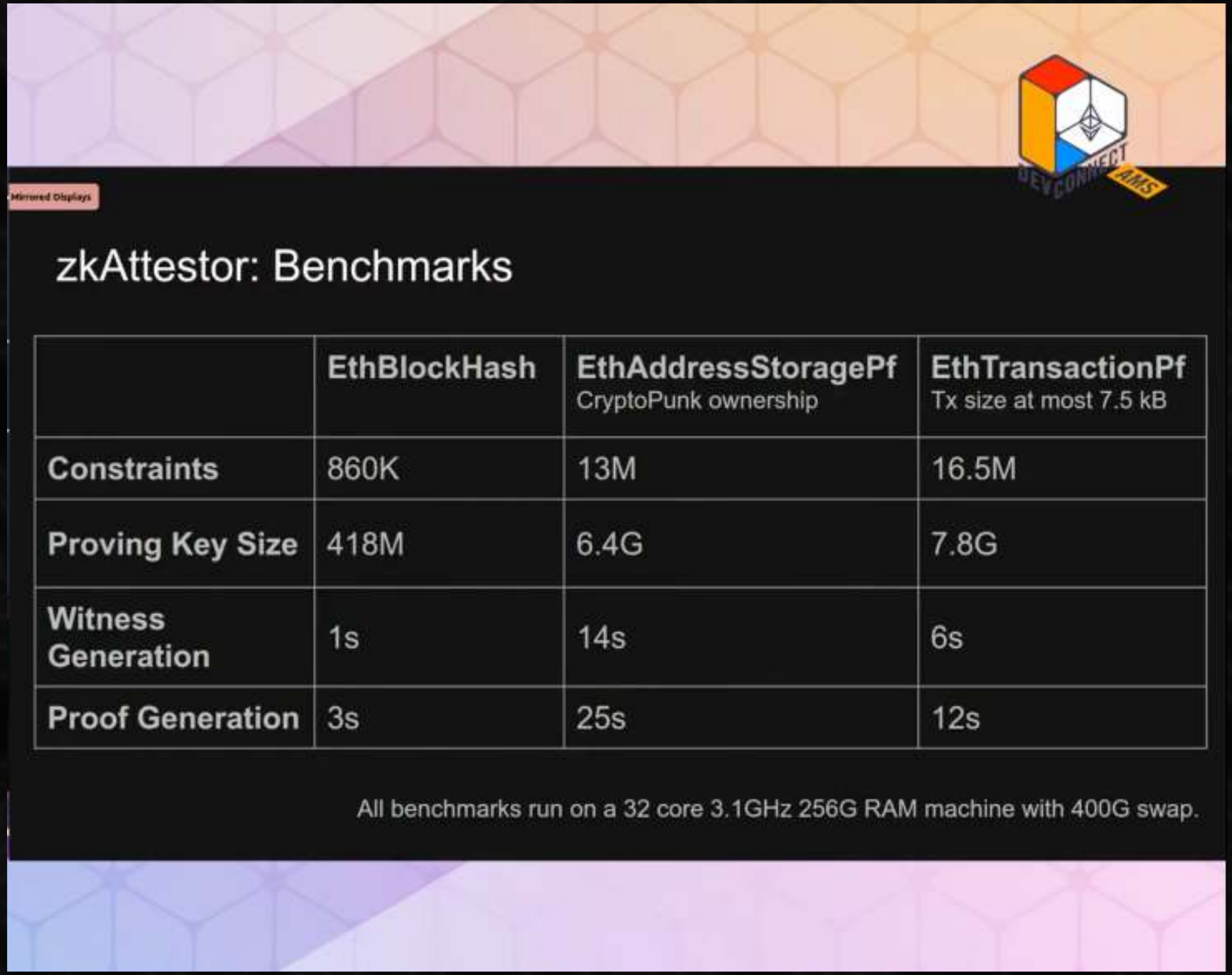
rapidsnark

rapid snark is a zkSnark proof generation written in C++ and intel assembly. That generates proofs created in circom and snarkjs very fast.

https://github.com/iden3/rapidsnark

Example 2: Rapid RapidSnark

- A popular tool chain for ZK applications
- Optimized Groth 16
  - BN254
  - 4 MSMs with G1, 1 MSM with G2, rest are NTTs
- Dev-ops: Beefy CPU instance.
- Rapid RapidSnark
  - replace CPU with GPU/FPGA in the cloud (AWS)
  - same cost (2 USD/h) or less



zkAttestor: Benchmarks

	EthBlockHash	EthAddressStoragePf CryptoPunk ownership	EthTransactionPf Tx size at most 7.5 kB
Constraints	860K	13M	16.5M
Proving Key Size	418M	6.4G	7.8G
Witness Generation	1s	14s	6s
Proof Generation	3s	25s	12s

All benchmarks run on a 32 core 3.1GHz 256G RAM machine with 400G swap.

Example 2: Rapid RapidSnark

- A popular tool chain for ZK applications
- Optimized Groth 16
  - BN254
  - 4 MSMs with G1, 1 MSM with G2, rest are NTTs
- Dev-ops: Beefy CPU instance.
- Rapid RapidSnark
  - replace CPU with GPU/FPGA in the cloud (AWS)
  - same cost (2 USD/h) or less

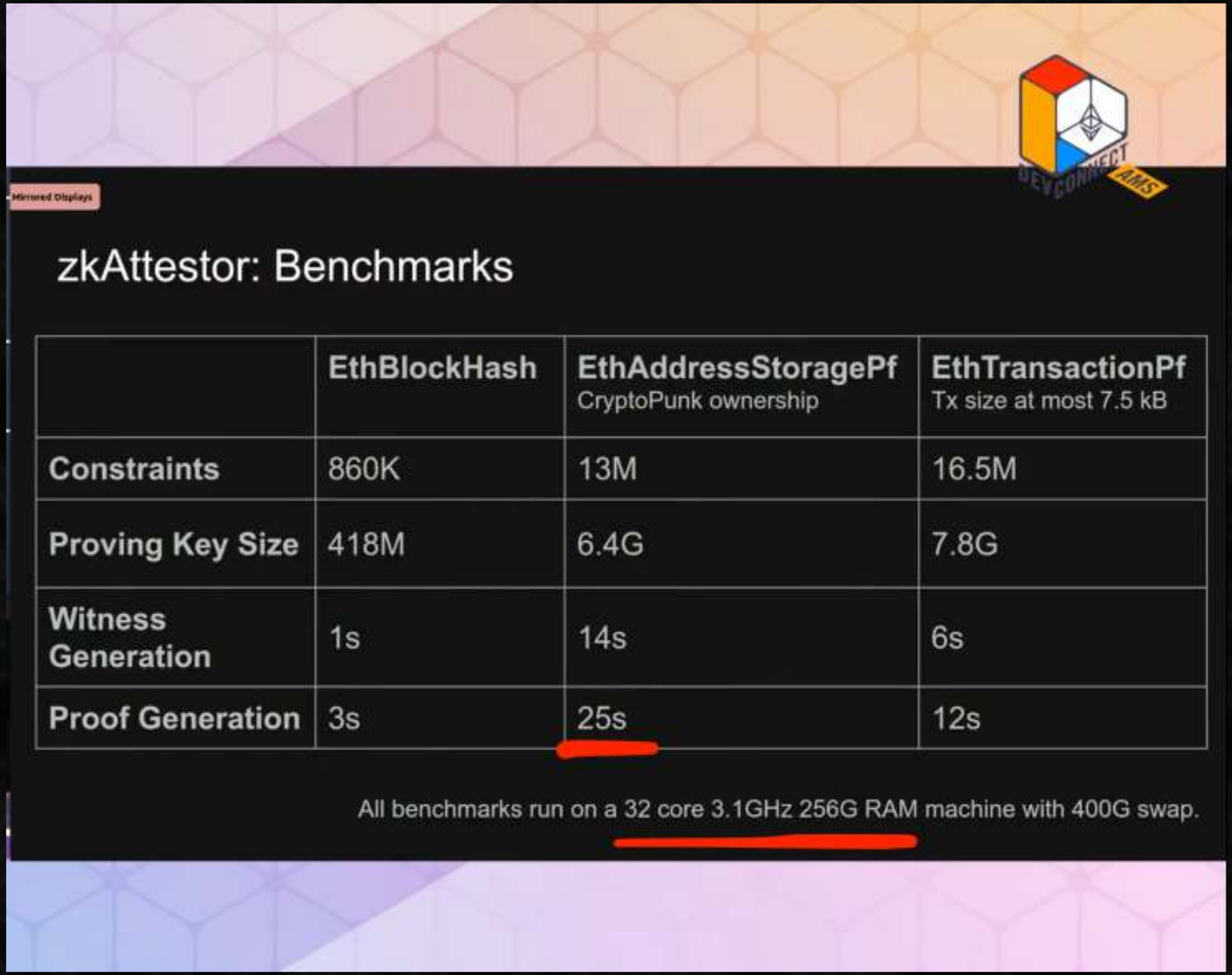
zkAttestor: Benchmarks

	EthBlockHash	EthAddressStoragePf CryptoPunk ownership	EthTransactionPf Tx size at most 7.5 kB
Constraints	860K	13M	16.5M
Proving Key Size	418M	6.4G	7.8G
Witness Generation	1s	14s	6s
Proof Generation	3s	25s	12s

All benchmarks run on a 32 core 3.1GHz 256G RAM machine with 400G swap.

Example 2: Rapid RapidSnark

- Accelerated: G1,G2,NTT
- Challenges:
  - Two elliptic curves
  - Hardware is not fully utilized
  - NTT grows $O(n \log(n))$, MSM grows $O(n)$
- Metric: Latency, Throughput
- Result: 5x improvement (open source soon)



zkAttestor: Benchmarks

	EthBlockHash	EthAddressStoragePf CryptoPunk ownership	EthTransactionPf Tx size at most 7.5 kB
Constraints	860K	13M	16.5M
Proving Key Size	418M	6.4G	7.8G
Witness Generation	1s	14s	6s
Proof Generation	3s	25s	12s

All benchmarks run on a 32 core 3.1GHz 256G RAM machine with 400G swap.

Thank you

INGONYAMA

Omer Shlomovits , ZKProof 5 Workshop