Zero Knowledge Machine Learning

Giving the blockchain eyes

Jason Morton

16 Nov 2022

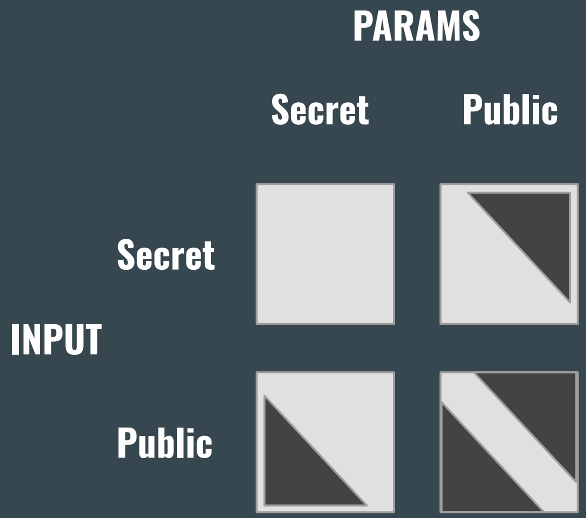
ZKP is to Digital Signatures as Ethereum is to Bitcoin

- Digital signatures: "I know secrets such that F(secrets, public inputs) = public outputs)," F a fixed function. This is how coins are spent.
- Zero-Knowledge Pf: "I know secrets such that F(secrets, public inputs) = public outputs)," F can be any program.
- Lets us move computation from the chain to the client.
- The most interesting program to put in a ZKP is a machine learning model. Effectively runs on-chain.

Jason Morton | ZKonduit
jason@zkonduit.com

Why ZKML?

- Gives the blockchain eyes to perceive the physical world.
- Makes it possible for a human, not a field element, to own digital assets.
- Lets smart contracts exercise judgement.
- Creates permissionless, unstoppable on-chain AI.



Jason Morton | ZKonduit
jason@zkonduit.com

EZKL: Turn an ONNX model into a zero-knowledge proof

- Prove + verify at command-line (or binary, contract, WASM)
- Adding layers daily, enough for small production models
- Performance improving 2-8x per month
- Focused on feature completeness, then optimization
- Apache 2.0 https://github.com / zkonduit / ezkl

node	opkind	output_max	min_cols	in_scale	out_scale	const_value	inputs	in_dims	out_dims	idx	bucket
#0 "input" Source	input	256	1	3	3				[3, 2, 2]	0	0
#1 "onnx::MatMul_1" Source	input	256	1	3	3				[3, 2, 2]	1	1
#2 "input.1" Source	input	256	1	3	3				[3, 2, 2]	2	2
#3 "conv2.bias" Const	const	1	1	3	3	[1...]			[3]	3	
#4 "onnx::Pow_18" Const	const	16	1	3	3	[16...]			[1]	4	
#5 "conv2.weight" Const	const	2	1	3	3	[-1...]			[3, 3, 2, 2]	5	

```python
def forward(self,x,y,z):
  x = self.sigmoid(self.connv2(x + y @ x**2 - self.relu(z))) + 2
  return x
```

x, y, z are tensors w/ runtime shape

node	opkind	output_max	min_cols	in_scale	out_scale	const_value	inputs	in_dims	out_dims	idx	bucket
#0 "input" Source	input	256	1	3	3				[3, 2, 2]	0	0
#1 "onnx::MatMul_1" Source	input	256	1	3	3				[3, 2, 2]	1	1
#2 "input.1" Source	input	256	1	3	3				[3, 2, 2]	2	2
#3 "conv2.bias" Const	const	1	1	3	3	[1...]			[3]	3	
#4 "onnx::Pow_18" Const	const	16	1	3	3	[16...]			[1]	4	
#5 "conv2.weight" Const	const	2	1	3	3	[-1...]			[3, 3, 2, 2]	5	
#6 "Pow_0" Pow	pow 2	65536	20	3	6		[0]	[3, 2, 2]	[3, 2, 2]	6	3
#7 "MatMul_1" MatMulInference	matmul	131072	31	3	6		[1, 6]	[2]	[3, 2, 2]	7	3
#8 "Add_2" Add	add	262144	31	6	6		[0, 7]	[3, 2, 2]	[3, 2, 2]	8	3
#9 "Relu_3" Clip	relu 1	256	12	3	3		[2]	[3, 2, 2]	[3, 2, 2]	9	4
#10 "Sub_4" Sub	sub	524288	31	6	6		[8, 9]	[3, 2, 2]	[3, 2, 2]	10	4
#11 "Conv_5" ConvHir	conv w/ padding: (2, 2), stride: (1, 1)	4194304	67	6	9		[10, 5, 3]	[3, 2, 2]	[3, 5, 5]	11	4
#12 "Sigmoid_6" Sigmoid	sigmoid 512	8	75	9	3		[11]	[3, 5, 5]	[3, 5, 5]	12	5
#13 "Constant_7" Const	const	16	1	3	3	[16...]			[1]	13	
#14 "Add_8" Add	add	32	92	3	3		[12, 13]	[3, 5, 5]	[3, 5, 5]	14	6

https://github.com/zkonduit/ezkl/

Jason Morton | ZKonduit
jason@zkonduit.com

```
Usage: ezkl [OPTIONS] --scale <SCALE> --bits <BITS>

Options:
Usage: ezkl [OPTIONS] <COMMAND>

Commands:
  table      Loads model and prints model table
  mock       Loads model and input and runs mock prover (for testing)
  fullprove  Loads model and input and runs full prover (for testing)
  prove      Loads model and data, prepares vk and pk, and creates proof, saving proof in --output
  verify     Verifies a proof, returning accept or reject
  help       Print this message or the help of the given subcommand(s)

Options:
  -S, --scale <SCALE>      The denominator in the fixed point representation used when quantizing [
  -B, --bits <BITS>        The number of bits used in lookup tables [default: 14]
  -K, --logrows <LOGROWS>  The log_2 number of rows [default: 16]
  -h, --help               Print help information
  -V, --version            Print version information
```

https://github.com/zkonduit/ezkl/

jason@zkonduit.com

```
jason@x:~/z/ezkl$ ./ezkl table -M examples/onnx_models/ff.onnx
[*]
 |
 |        EZKL
 |
 |
 |        -----------------------------------------------------------
 |        Easy Zero Knowledge for the Lyrical.
 |        -----------------------------------------------------------
 |
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| node                   | opkind   | output_max | min_cols | in_scale | out_scale | const_value | input_shapes | output_shapes   | in_dims | out_dims | hyperp
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| #0 "input" Source      | input    | 256        | 1        | 7        | 7         |             |              | [Some([1, 3])]  |         | [1, 3]   |
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| #1 "fc1.weight" Const  | const    | 71         | 1        | 7        | 7         | [71...]     |              | [Some([4, 3])]  |         | [4, 3]   |
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| #2 "fc1.bias" Const    | const    | 60         | 1        | 7        | 7         | [-60...]    |              | [Some([4])]     |         | [4]      |
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| #3 "Gemm_0" Gemm       | affine   | 54528      | 4        | 7        | 14        |             |              | [None]          | [3]     | [4]      |
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
| #4 "Relu_1" Clip       | relu 128 | 426        | 4        | 14       | 7         |             | [Some([4])]  | [Some([1, 4])]  | [4]     | [4]      |
+------------------------+----------+------------+----------+----------+-----------+-------------+--------------+-----------------+---------+----------+------
jason@x:~/z/ezkl$
```

https://github.com/zkonduit/ezkl/

jason@zkonduit.com

```
jason@x:~/z/ezkl$ ./ezkl prove -M examples/onnx_models/ff.onnx -D examples/onnx_models/ff_input.json -O thepf.pf
[*]
 |
 |
 |
 |
 |
 |
 |
 |          ------------------------------------------------------------
 |          Easy Zero Knowledge for the Laconic.
 |          ------------------------------------------------------------
 |
[*] Proof with ipa
[*] loading data from examples/onnx_models/ff_input.json
[*] public input length (network output) 4
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
[*] number of advices used: 9
[*] configuring model
[*] model layout
[*] VK took 3
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
[*] number of advices used: 9
[*] configuring model
[*] model layout
[*] PK took 1
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
[*] number of advices used: 9
[*] configuring model
[*] model layout
[*] Proof took 7
```

https://github.com/zkonduit/ezkl/

jason@zkonduit.com

```
jason@x:~/z/ezkl$ ./ezkl verify -M examples/onnx_models/ff.onnx   -P thepf.pf
[*]
 |
 |          EZKL
 |
 |
 |          -----------------------------------------------------------
 |          Easy Zero Knowledge for Layers.
 |          -----------------------------------------------------------
 |
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
[*] number of advices used: 9
[*] configuring model
[*] model layout
[*] loading model from examples/onnx_models/ff.onnx
[*] quantizing model activations
[*] number of advices used: 9
[*] configuring model
[*] model layout
Verify took 0
Verified: true
```

https://github.com/zkonduit/ezkl/

jason@zkonduit.com

Timeline

- July: MNIST in snark w/ lookup nonlinearities
- Aug: EVM verification (single-model)
- Sept: tensors, ONNX, refactoring, quantization: make general purpose, devex
- Oct: new ops, docs, display, subcommands, pf serialization, auto fusing ops, devex
- Nov: Pf sys abstraction, EVM for general models, new ops, fewer passes, flexible quantization, tolerance, devex
- Optimization, recursion/composition, execution environments

Jason Morton | ZKonduit
jason@zkonduit.com

Quantization

- Native weights and activations are small 32-bit floats. Quantization to field elements is different from performance-oriented int8 quantization, although there is overlap.
- i8 to i32 fixed-point representation, trade-off w/ zk time/space performance
- For single models, manual quantization works and a lot of shortcuts can be taken, but for general models an automated quantization strategy is needed and can be a bit fun.
- Numerical errors have to be anticipated and handled.
- We take a vaguely type-theoretic approach now, may move to a more statistical approach.
- This is likely to be an area for significant mathematical/numerical innovation.

Jason Morton | ZKonduit
jason@zkonduit.com

ZKML allows scalable automated oracles

Signed ingestion

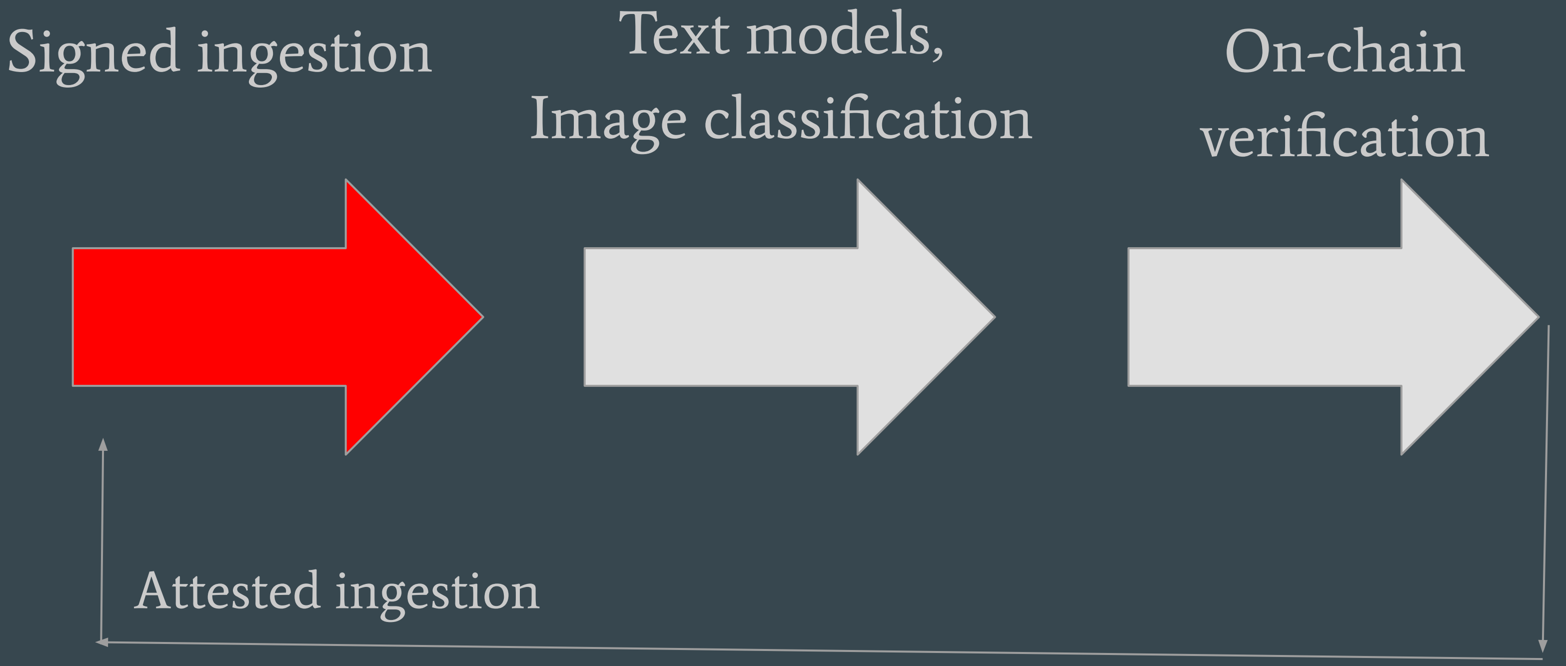
Text models,
Image classification

On-chain
verification

Attested ingestion

Jason Morton | ZKonduit
jason@zkonduit.com

ZKML allows scalable automated oracles



Jason Morton | ZKonduit
jason@zkonduit.com

Authenticated content is here and verifiable in a zk-snark

- Http: SXG, signed AMP, signed endpoints (Cloudflare one-click SXG, nginx)
- Email (DKIM)
- Images at the publisher: C2PA, Images at the camera
- Third-party notaries (Lit, TLS Notary, Deco)

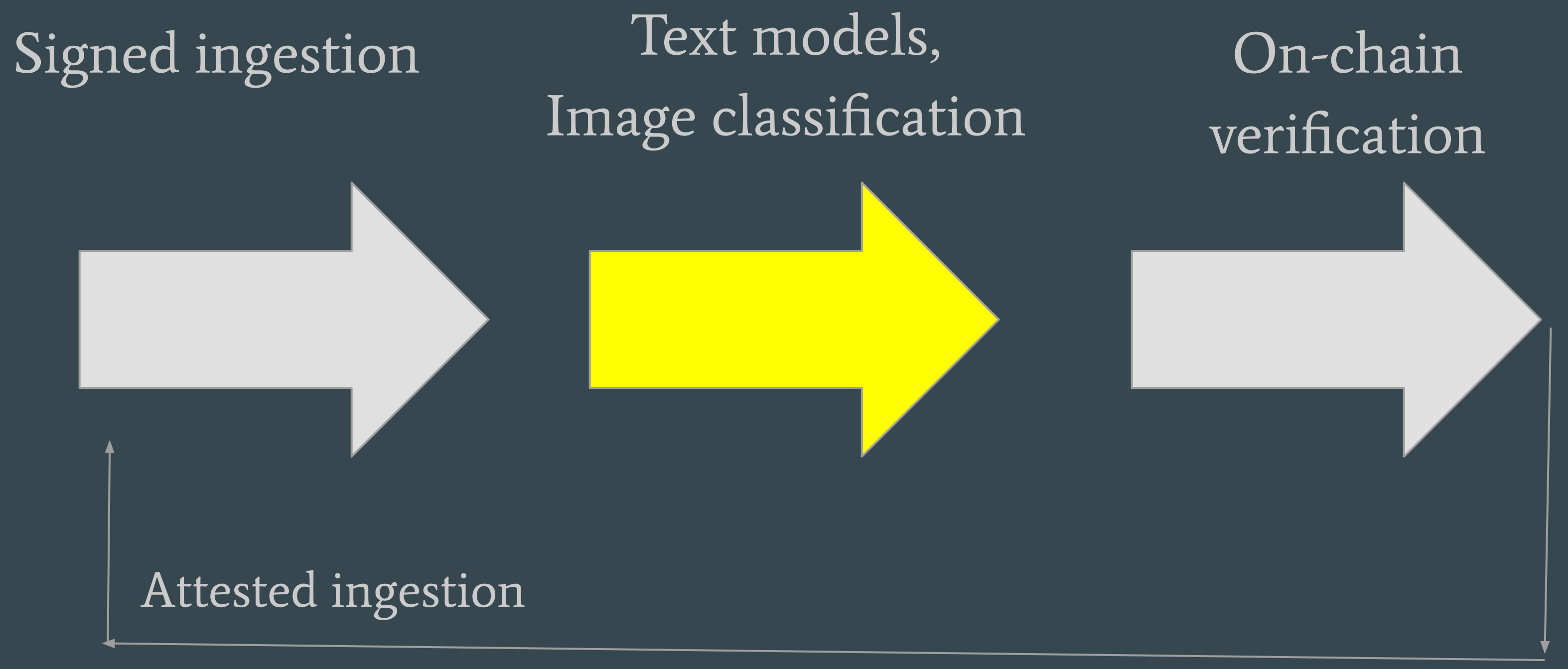
All use standard signature schemes (ECDSA, RSA, Ed25519) that are now verifiable in zk-snarks and/or on-chain

We need an https-like push to SIGN YOUR DATA

Jason Morton | ZKonduit
jason@zkonduit.com

Content Authority Initiative Members

Adobe
AFP
AP
arm
BBC
camera bits
CBC Radio-Canada
dpa
ELL ERNST LEITZ LABS
GANNETT
gettyimages
McCLATCHY
METAPHYSIC
Microsoft
Nikon
The New York Times
NVIDIA
Qualcomm
reface
rs
REUTERS
SMARTFRAME
stern
Synthesia
THE GLOBE AND MAIL
Truepic
VII



Jason Morton | ZKonduit
jason@zkonduit.com

Ontogeny recapitulates phylogeny roadmap

With ONNX compilation, we now

- Download the next model in the history of AI (from MNIST to Stable Diffusion)
- Fix any model size or quantization problems, implement any new nodes or gadgets
- Repeat

Jason Morton | ZKonduit
jason@zkonduit.com

Scaling with five tools

- Optimization and tuning: layout, care with memory usage, avoiding repeated work, etc.
- Aggregation unlocked: compress, split proof generation 'horizontally'.
- Recursion coming soon: split proof generation 'vertically'
- Fusion and abstraction => new arguments w/ smaller overhead.
- Composition: mix proof systems

Jason Morton | ZKonduit
jason@zkonduit.com

Aggregation

Easy to make,
Hard to verify
Proofs

Aggregator /
Compressor

Hard to make,
Easy to verify
Proof

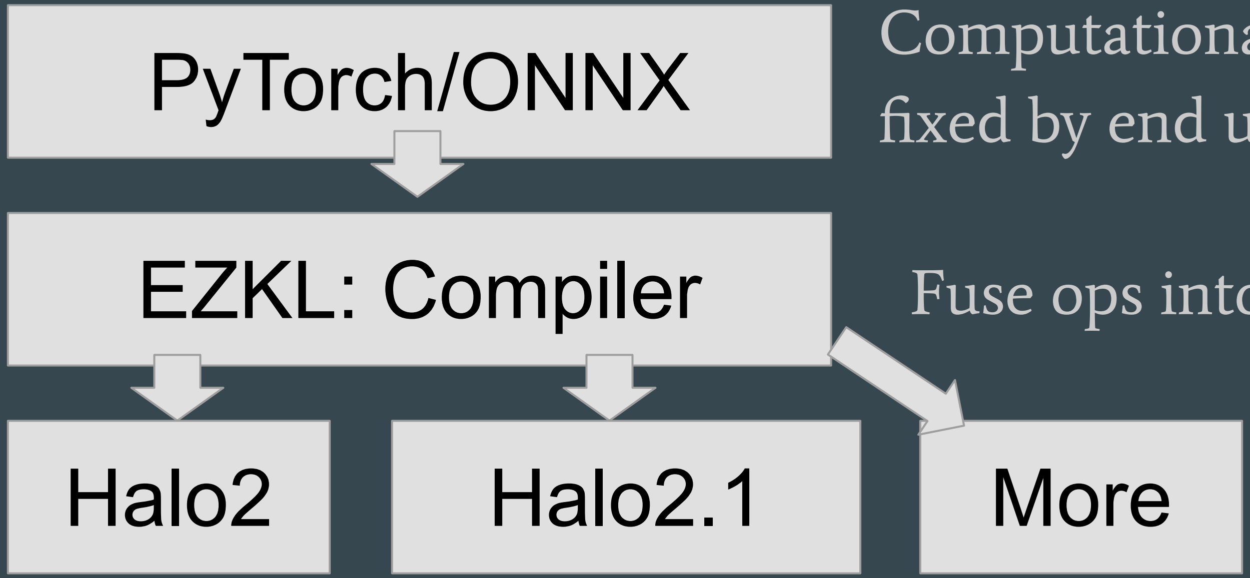
Jason Morton | ZKonduit
jason@zkonduit.com

Advantages of PyTorch / Tensorflow / ONNX

- High-level representation of a circuit, unlike most proposed intermediate representations for proof systems.
- e.g. matrix multiplication, not just a series of polynomial constraints.
- Perhaps the most widely used and familiar circuit notation in the wild.
- Easy onboarding for people with AI/ML/Data Science backgrounds

Jason Morton | ZKonduit
jason@zkonduit.com

Fusion and abstraction



Computational graph,
fixed by end user

Fuse ops into supported args

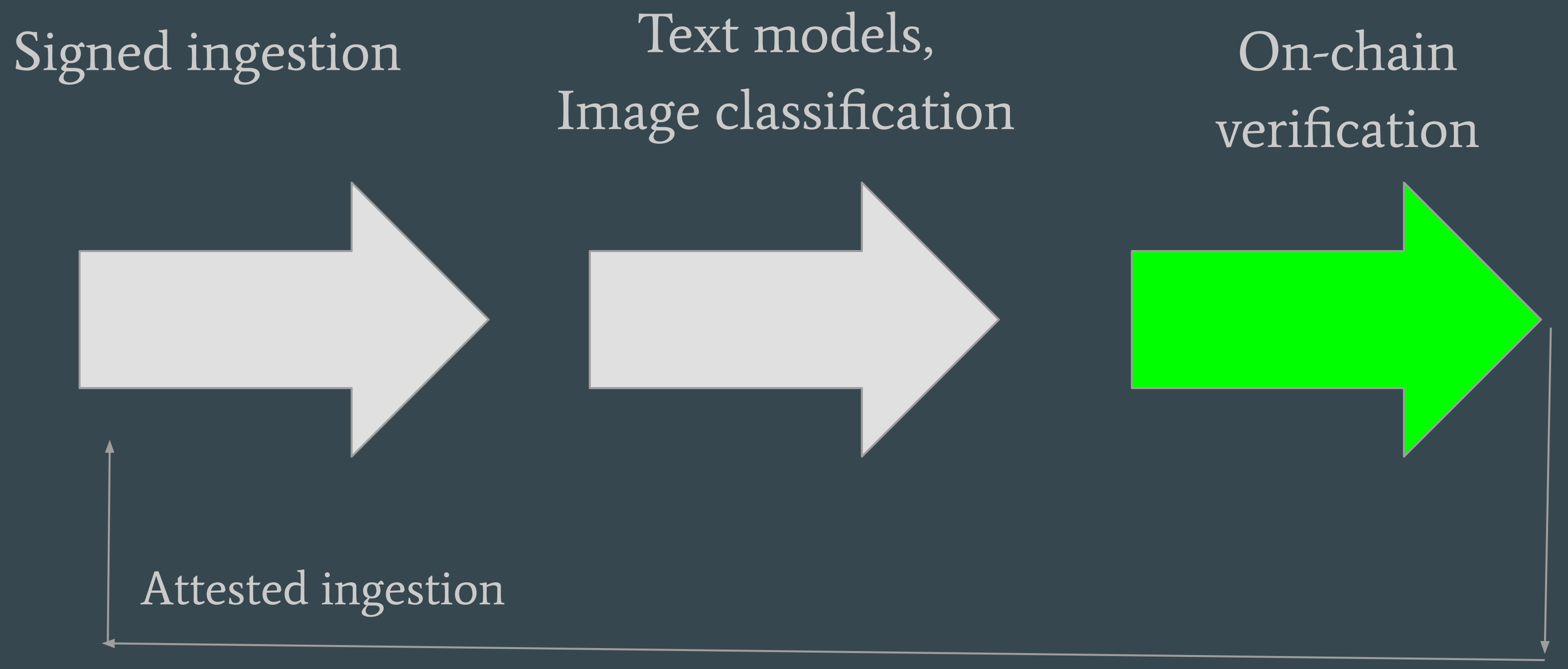
Fast-changing proof systems as compile targets

Jason Morton | ZKonduit
jason@zkonduit.com

Abstraction and Fusion

- Change the cryptographic backend but use the same high-level graph description file (onnx)
- We have access to deeper understanding of intent (e.g. matmul, not a bunch of equations), can change the argument
- User only sees performance improvement

Jason Morton | ZKonduit
jason@zkonduit.com



Jason Morton | ZKonduit
jason@zkonduit.com

Easy to make,
Hard to verify
Proofs

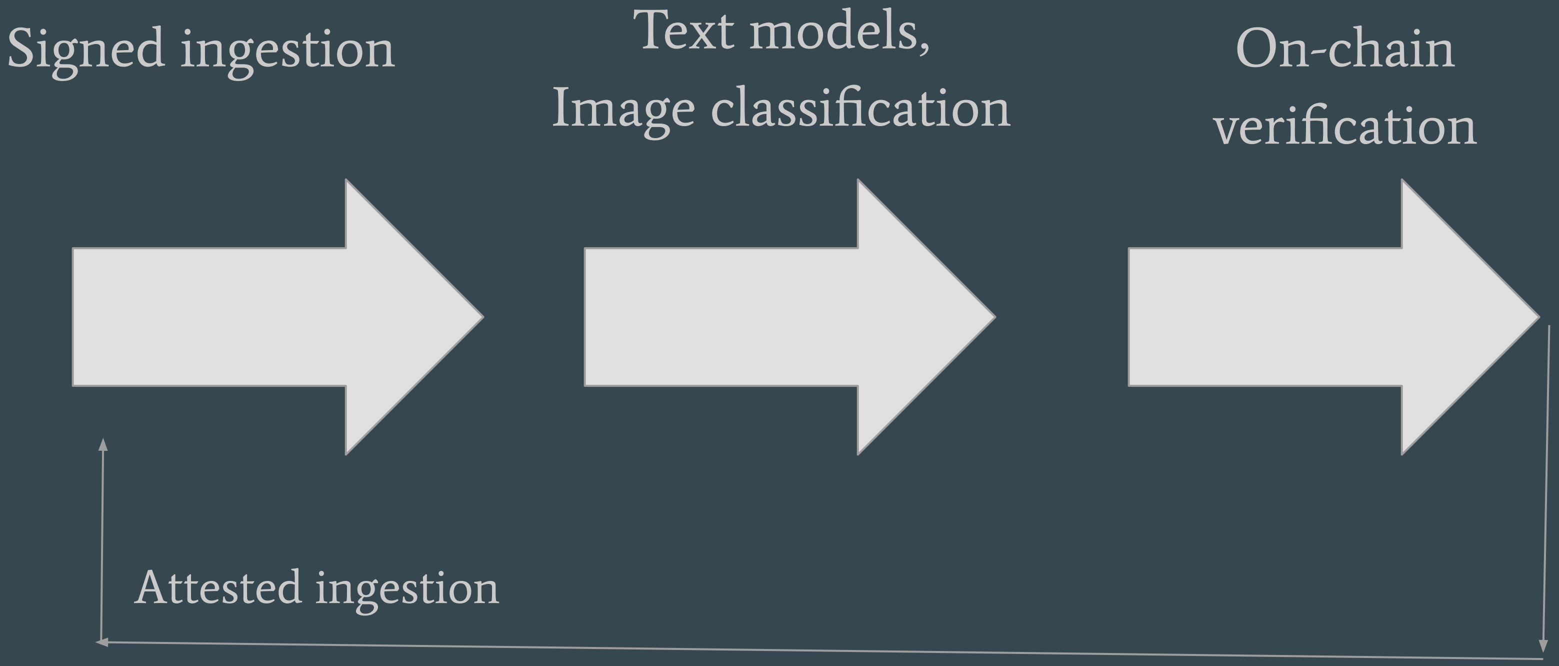
Aggregator /
Compressor

Hard to make,
Easy to verify
Proof

```
pk
deployment code
agg pf
evm verify
[src/main.rs:228] result.gas = 608421
jason@x:~/z/halo2/dlverified$
```

Jason Morton | ZKonduit
jason@zkonduit.com

Scalable trustless on-chain data feeds



Jason Morton | ZKonduit
jason@zkonduit.com

ZKML will be table stakes for chains

- Delivering on the promise of blockchain to a mass audience will require more robust but still-decentralized identity solutions
  - Full identity solutions years off, but growth will be fast
- ZKML Oracles will be simpler, faster, and much more scalable
  - Put arbitrary off-chain data on chain
  - Opens the firehose to get data on chain
- A ZKML model is a ‘smart judge’ that can interpret ambiguous events...

Such as

ZK KYC

- Prove the person and id match, and the id is not sanctioned
- Regulators won’t accept as KYC, but
- Could have prevented tornado sanctions

Jason Morton | ZKonduit
jason@zkonduit.com

Prediction markets

- Classifying text into few classes possible with small models
- Construct a smart contract that pays if a news story classifies to the predicted outcome (election outcome, hurricane intensity, covid variant)
- Anyone can download signed story, run model, submit proof

Jason Morton | ZKonduit
jason@zkonduit.com

Gut checks for smart contracts

- Smart contract or abstracted account adds a zkml fraud / spam check for unusual behavior

Put the A in DAO

- Now: humans judge, vote, signatories use multisig
- Replace with on-chain AI automation, e.g. for contract fulfillment

AI Bias

- Prove performance and unbiasedness of a committed model without revealing the parameters
- E.g. resume screening, TSA.
- Many AI fairness and audit laws in the pipeline in US and EU

MPC+ZK: Genetic screening

- Patient wants a prediction (e.g. chance of developing a genetic disorder), but to check anonymously, choose whether and to whom to reveal
- Screening models are trained on controlled data, and cannot be publicly shared
- Model should be private to model owner, data to patient
- Model owner and patient can prove inference in MPC, revealing only outcome, and patient gets a certified prediction

Differential Privacy + ZKML: Census

- Secret real data is committed to publicly (revealing nothing) at regular intervals
- Server creates differentially-private noisy marginal summary on which clients are free to prototype analysis
  - Client iterates locally on summary, decides on model M and sends to data owner
- Server runs the model in ZK on the real full-table data, returns the result to the client, and proves to the client:
  - The real data matches the commitment
  - The real data was used to create the noisy marginal summary
  - When model M was run on the real data, it produced the returned result

Thank you!