The Verifiability Vision

Devcon, Bangkok 2024

Jens Groth
Chief Scientist, Nexus - nexus.xyz

Proof carrying data

Vision: all data is computed from verified inputs and comes with proof of provenance

Statements and witnesses

- This image depicts Lake Ontario on September 7, 2024
  - Witness: raw data & GPS coordinates signed by camera
- This ciphertext encrypts a valid vote
  - Witness: vote and randomness used to encrypt it
- The customer is over 18 years old
  - Witness: signed data from passport chip
- The smart contract received a deposit
  - Witness: blockchain state and evidence of finalization

Witness may be confidential

Proof system for a class of statements

Setup: (pk, vk)

Prover (pk, stmt, wit) $\rightarrow \pi$

Verifier (vk, stmt, π) $\rightarrow$ yes/no

- **Complete** : honest prover with witness to truth of statement can create proof π
- **Sound**: evil prover cannot create valid proof π for a false statement
- **Zero knowledge**: the proof π reveals nothing about the witness

Challenges to make proof carrying data take off

- Versatility and ease of use
- Cost
- Composability
- Trust anchors in the real world

Versatility and ease of use

Ease of use

- Hire a cryptographer

- “ASIC” approach: use tools to generate a purpose-built circuit

- “CPU” approach: compile to a zkVM

Zero Knowledge Virtual Machine

- Program P specified according to the Instruction Set Architecture
- Input x to the program (public)
- Auxiliary input w to the program (maybe not public)
- Output y
- Proof π

Zero Knowledge Virtual Machine & matching verification algorithm

Developer experience

```
$ cargo nexus prove
$ cargo nexus verify
```

```rust
#![no_std]
#![no_main]

fn fib(n: u32) -> u32 {
    match n {
        0 => 0,
        1 => 1,
        _ => fib(n - 1) + fib(n - 2),
    }
}

#[nexus_rt::main]
fn main() {
    let n = 7;
    let result = fib(n);
    assert_eq!(result, 13);
}
```

Cost

SNARK

Setup: (pk, vk)

Prover (pk, stmt, wit) $\rightarrow \pi$

Verifier (vk, stmt, π) $\rightarrow$ yes/no

- **Succinct**: proof π is small regardless of how complex the statement is (cheap)
- **Non-interactive**: prover creates proof π by herself (copy and distribute)
- **Argument of Knowledge**: proof implies witness could be constructed (no false stmt)

Succinct proof

- The proof π is small
- Verification time is independent of the program's execution time T and $|w|$

Verification is cheaper than computation

Prover efficiency is the remaining bottleneck

- Proof size ✓
- Verification time ✓
- Proving time ?

As low as 200 bytes

As low as reading the statement

$$\text{Overhead} = \frac{\text{cost to prove}}{\text{cost to compute}}$$

Many orders of magnitude :-(

Prover efficiency is improving

- CPUs: #transistors per IC increases ~50%/year (Moore's law)
- zkVMs: #proved CPU cycles per second increases >1000%/year (in 2024)

Polite language amongst cryptographers when describing prover time

Applications in the blockchain space

- **Scalability**
  - **Roll-ups:** increasing throughput in slow blockchains
  - **Bridges**: cross-chain interaction
  - **Co-processors**: increasing processing speed or decreasing cost
  - **L1s**: compressing blockchains
- **Privacy**
  - **L1s**: private transactions
  - **Smart contracts**: private execution

Why is blockchain willing to pay the large proving cost?

1. Unlocks unique capability
2. Replicated verification may be even more expensive
3. Cost of storage of original data too expensive

zkRollups: increasing throughput

- On-chain execution: low throughput, expensive

- Off-chain execution & on-chain verification: higher throughput, cheaper

Applications in the blockchain space

- **Scalability**
  - **Roll-ups:** increasing throughput in slow blockchains
  - **Bridges**: cross-chain interaction
  - **Co-processors**: increasing processing speed or decreasing cost
  - **L1s**: compressing blockchains
- **Privacy**
  - **L1s**: private transactions
  - **Smart contracts**: private execution

Why is blockchain willing to pay the large proving cost?

1. Unlocks unique capability
2. Replicated computation may be even more expensive
3. Cost of storage of original data too expensive

Composability

Proof carrying data

Succinctness ensures recursive proofs of proofs of proofs ... to stay small

Composability for verifiable data

- Direct recomputation: evidence size explosion
- Interactive techniques, e.g., spot-checking: impossible to recurse
- Economic security, e.g., slashing if cheating detected: hard to track risk over paths of statements

SNARKs compose
- Succinct: no size explosion
- Non-interactive: proofs can be used as input to further computation
- Arguments of knowledge: ~100% soundness so no degradation

Amortizing proof costs over many verifications

Proof indirectly verified many times, since it supports all subsequent proofs

Trust anchors and trust management

The oracle problem of zkp

Trust anchors

Camera signs raw image, Ethereum finalizes block n, President signs law,...

Proof management

What happens if a trust anchor fails?

What success looks like

Blockchain: proof singularity

L3s	L2s	L1
tx_1, tx_2, tx_3	$\Delta_1, \pi_1, \Delta_2, \pi_2, tx_7$	Δ, π
tx_4, tx_5, tx_6		
tx'_1, tx'_2	$\Delta'_1, \pi'_1, \Delta'_2, \pi'_2, tx'_6$	Δ', π'
tx'_3, tx'_4, tx'_5		

Action

Verification & settlement

Digital world: web of verifiable data