Fuzzing Zero-Knowledge Infrastructure

Valentin Wüstholz

Consensys Diligence

Joint work with

Christoph Hochrainer, Anastasia Isychev, and Maria Christakis (TU Vienna)

Definition: zero-knowledge (ZK) infrastructure 🤓

Software components that are used for compiling, executing, proving, or verifying ZK circuits.

Examples: processing pipelines for ZK circuits and zkEVMs

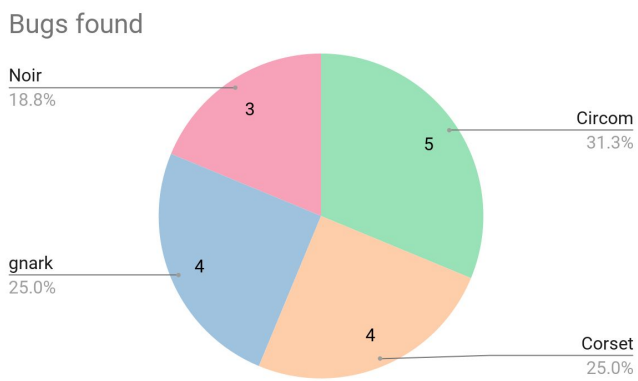
Why?

- ZK infrastructure is **highly complex (“moon math”)** and **highly critical** for the correct operation of several L2 chains

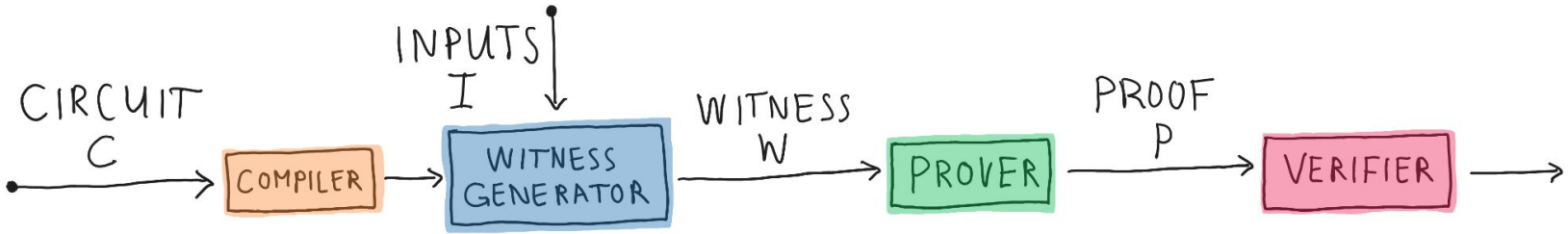
  ⇒ **Bugs can result in catastrophic financial and reputational damage**
- We have not seen a catastrophic incident, comparable to the TheDAO hack in 2016, but it **requires great engineering discipline and rigorous testing** to bulletproof such software components, given their inherent complexity
- **Fuzzing is widely used in industry** (Microsoft, Google, Meta, etc.) to catch bugs before hackers can exploit them

Circuzz

- **Goal: find critical bugs in processing pipelines for ZK circuits**
- **Example pipelines/DSLs:** Cairo, Circom, Corset, gnark, and Noir
- **Already supports 4 pipelines:** Circom, Corset, gnark, and Noir
- Found **16 bugs** so far (15 fixed) 😎

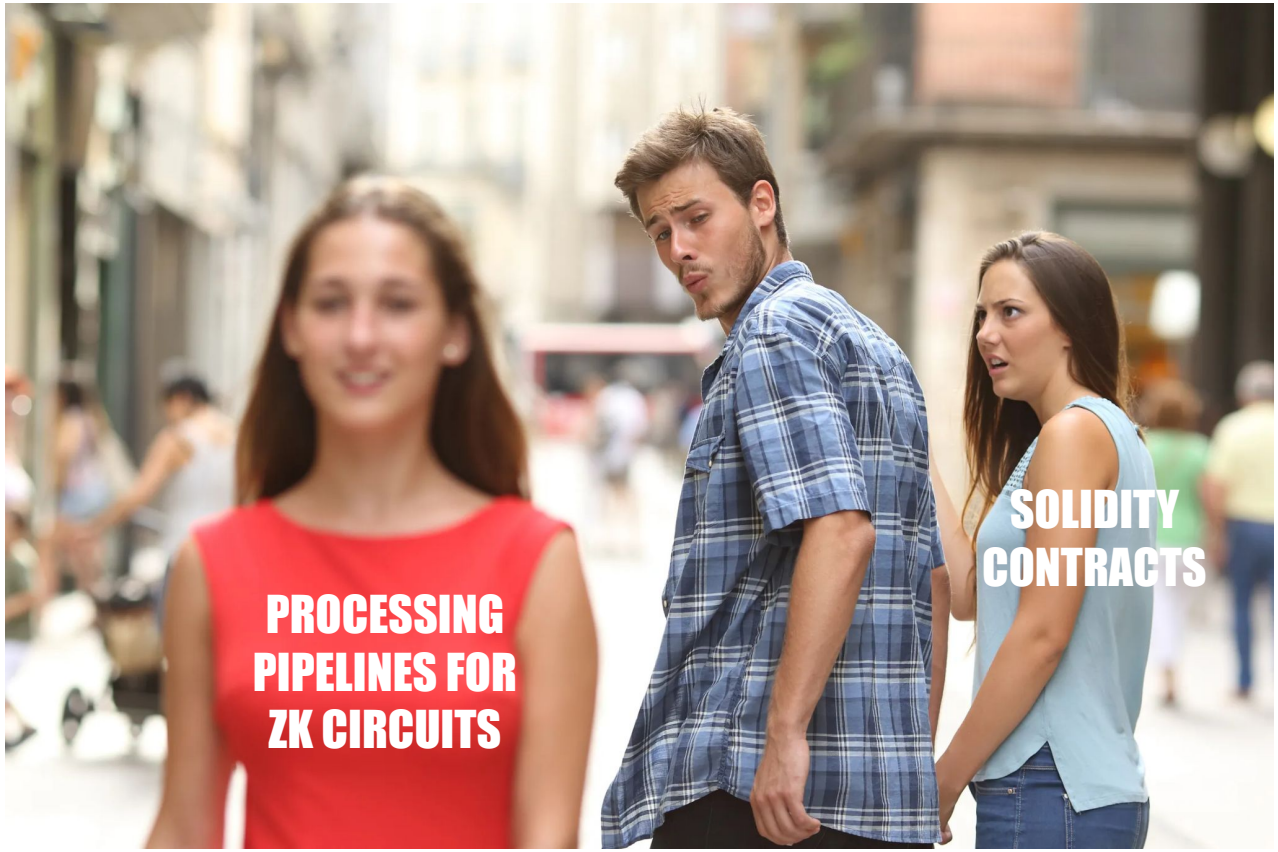


A typical processing pipeline



How do we find these bugs?

- **We are not cryptographers!** 😅
- We just treat each **pipeline as a black box** (like users, not like developers)
- **How can we still find critical bugs?!?** 🤔
- We have years of experience with building fuzzers for complex domains:
  - **Smart contracts** (VMCAI ‘22, ICSE ‘20, FSE ‘20)
  - **ML models** (AAAI ‘24, IJCAI ‘23, ISSTA ‘22, ICAPS ‘22)
  - **Program analyzers** (ASE ‘24, ASE ‘24, ISSTA ‘23, FSE ‘21, FSE ‘20, ISSTA ‘19)
- We have a not-so-secret weapon: **Metamorphic Testing**


PROCESSING
PIPELINES FOR
ZK CIRCUITS
SOLIDITY
CONTRACTS

Metamorphic Testing: how to elegantly define test oracles

- **Ex. 1:** How do you test a sorting function Sort(X)?

  Sort(X) == Sort(RandomShuffle(X))

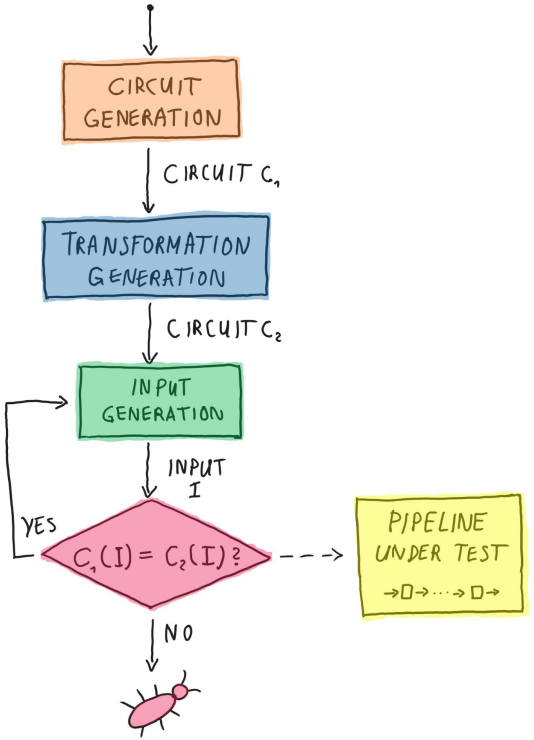
- **Ex. 2:** How do you test a shortest-path computation ShortestPath(N, M, G)?

  ShortestPath(N, M, G) <= ShortestPath(N, M, RemoveRandomEdge(G))

- **Ex. 3:** How do you test a compiler?

  Run(Compile(P)) == Run(Compile(AddRandomDeadCode(P)))

How do you test processing pipelines?

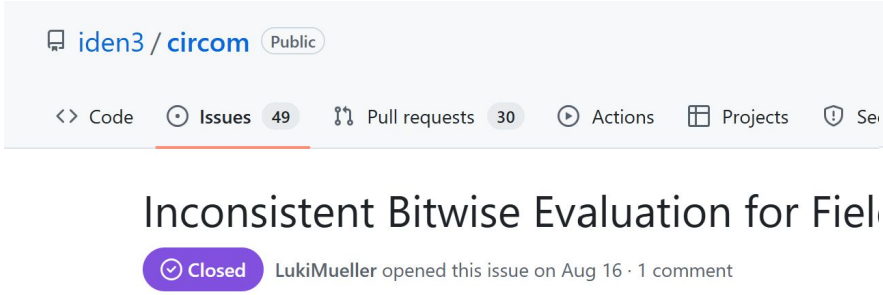


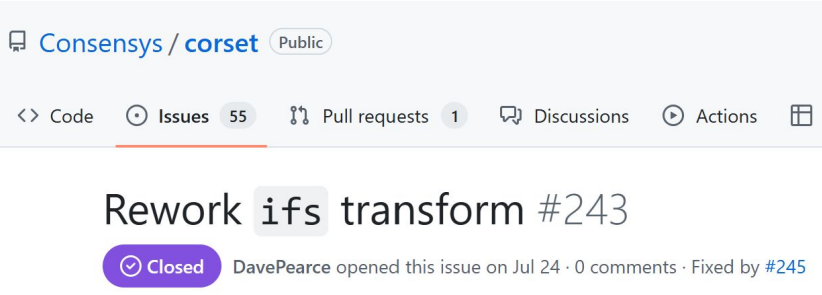
Example transformations

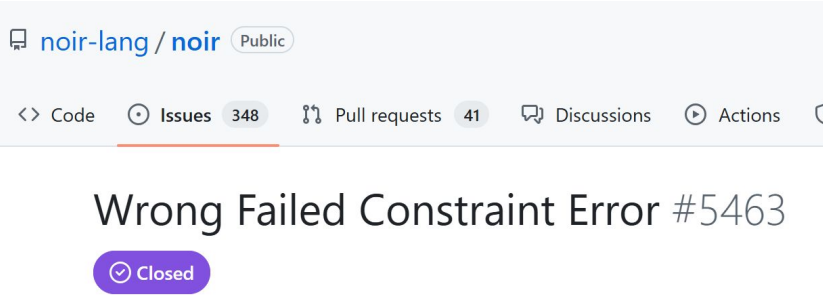
- e ⇒ (e * 1)
- e ⇒ (e / 1)
- e ⇒ (-(-e))
- e1 * e2 ⇒ e2 * e1
- e ⇒ ((e - $r) + $r) ($r is fresh random expression)
- …

currently 87 supported transformation rules

Example bugs







Example bug 1 (Circom #288)

```
template t1() {
  signal input in0, in1;
  signal output out1;
  out1 <-- (~ p);
  assert(in0 != in1);
}
```

```
template t2() {
  signal input in0, in1;
  signal output out2;
  out2 <-- (~ (((1 - 0) / 1) * p));
  assert(in0 != in1);
}
```

out1 != out2

p = 21888242871839275222246405745257275088548364400416034343698204186575808495617

Example bug 2 (gnark #1227)

```
func (circuit *C1) Define(api frontend.API) error {
  api.AssertIsLessOrEqual(1, 0)
  return nil
}
```

```
func (circuit *C2) Define(api frontend.API) error {
  api.AssertIsLessOrEqual(1, api.Or(0, 0))
  return nil
}
```

C1: no witness
C2: witness

Takeaway

- You don’t need to be a cryptographer to find bugs in crypto components!
- So far, we just scratched the surface
- We need continuous fuzzing!

  (already implemented for gnark and Corset coming soon)
- Reach out to fuzz your ZK infrastructure!

Fuzzing Processing Pipelines for Zero-Knowledge Circuits

CHRISTOPH HOCHRAINER, TU Wien, Austria
ANASTASIA ISYCHEV, TU Wien, Austria
VALENTIN WÜSTHOLZ, ConsenSys, Austria
MARIA CHRISTAKIS, TU Wien, Austria

Zero-knowledge (ZK) protocols have recently found numerous practical applications, such as in authentication,