Formal Verification for ZK Circuits

Yanju Chen & Junrui Liu
University of California, Santa Barbara & Veridise Inc.
Contents

Overview
Verification Techniques Spectrum
Motivations, Challenges

01

Formal Verification for Core Library Circuits
Highest Security Guarantee
Interactive Theorem Proving

02

Formal Verification for Application Circuits
Automated Methods
Symbolic Execution, Abstract Interpretation

03

Plans & Next Steps

04
Overview
Verification Techniques Spectrum

**Testing**
- Pros: Less Human Efforts
- Cons: Less Guarantee

**Static Analysis**

**Formal Verification**
- Pros: Highest Guarantee
- Cons: More Human/Expert Efforts
ZK Circuit Verification

Why?
- Application Security
- Library Reusability
- ...

Challenges?
- Scalability
- Coverage
- Extensibility
- ...

What?
- Functional Correctness
- Uniqueness Property
- ...

Smart Contracts

Application Circuits

Core ZK Libraries & Building Blocks
ZK Circuit Verification: Current Roadmap

Core Library Circuits (Manual)
- A tiny but critical and frequently used set of circuit building blocks (e.g., circomlib)
- Formal verification using interactive theorem proving
- This will provide the highest guarantee, but requires manual/expert efforts

Application Circuits (Automated / Semi-Automated)
- Majority of the application circuits belong to this category
- Automatically translating program into machine checkable formula
- Abstract level static analysis to over-approximate the range of each variable/wire
Functional Correctness
"Do the constraints correctly represent user intent?"

int $x[3]; y = x[j]$  "$y$ should be sampled from array $x$"

```
pragma circom 2.0.0;
template test() {
    signal input $x[3], j$;
    signal output $y$;
    signal $i0, i1, i2$;
    signal $y0, y1, y2$;
    $i0 <-- j==0? 1:0$;
    $i0 * (j-0) === 0$;
    $i1 <-- j==1? 1:0$;
    $i1 * (j-1) === 0$;
    $i2 <-- j==2? 1:0$;
    $i2 * (j-2) === 0$;
    $y0 <= i0*x[0]$;
    $y1 <= i1*x[1]$;
    $y2 <= i2*x[2]$;
    $y <= y0 + y1 + y2$;
}
component main = test();
```

```
pragma circom 2.0.0;
template test() {
    signal input $x[3], j$;
    signal output $y$;
    signal $i0, i1, i2$;
    signal $y0, y1, y2$;
    $i0 <-- j==0? 1:0$;
    $i0 * (j-0) === 0$;
    $i1 <-- j==1? 1:0$;
    $i1 * (j-1) === 0$;
    $i2 <-- j==2? 1:0$;
    $i2 * (j-2) === 0$;
    $y0 <= i0*x[0]$;
    $y1 <= i1*x[1]$;
    $y2 <= i2*x[2]$;
    $y <= y0 + y1 + y2$;
}
component main = test();
```

```
y = x[0] || y = x[1] || y = x[2]
```
Query/Specification
Formal Verification for Core Library Circuits (Junrui)
Formal Verification for Application Circuits
Vulnerability/Bug Detection in Application Circuits

Application Circuits
- Large (>5000 LOC, millions of constraints)
- Contains non-library constraints

What to Verify / Sources of Bugs
- Functional Correctness
  "I think what I wrote is all I want! ... no?"
- Uniqueness Property
  "I think I've already included all range checks! ...probably?"
Automated Verification of Functional Correctness

Automated Verification Techniques
- Symbolic Execution
- Abstract Interpretation
- ...

Picus is based on symbolic execution. (https://github.com/chyanju/Picus)
Symbolic Execution

An interpreter follows the program, assuming symbolic values for inputs rather than obtaining actual inputs as normal execution of the program would.

Example:

```
int x[3]; y = x[j]
```

Auto-Generated SMT Constraints

```
(\&\& (\leq 0 j) (< j 3))
(ite* (\Rightarrow (= 0 j) x1) (\Rightarrow (= 1 j) x2) (\Rightarrow (= 2 j) x3))
```

"y should be sampled from array x"

Query/Specification

```
y = x[0] || y = x[1] || y = x[2]
```

Constraint Solver

(Verified)

Symbolic Compilation

(e.g., )

Interpreter

Symbolic Evaluator

Program

Symbolic Input

Constraints

Solver

Solution
Weak Verification (IO Uniqueness)
- This tests if, given the input variables in a QAP, the output variables have uniquely determined values.
- Example: $x[1], x[2], x[3]$ and $j$ are fixed, $y$ is queried.

Witness Uniqueness
- This tests if all the witness variables that appear in all equations, and not just input and output variables, collectively are uniquely determined.
- Example: $x[1], x[2], x[3]$ and $j$ are fixed, $i_1, y_1$ and $y$ are queried.

Strong Uniqueness
- This tests if the QAP is exactly equivalent to a formal mathematical specification.
- (Similar to function correctness)
Automated Verification of Uniqueness Property

Related Work: Ecne (https://github.com/franklynwang/EcneProject)
- Ecne is based on a worklist + fixed point algorithm
- Needs manually devised inference rule for deducing uniqueness
- Applies well to circuits within inference scope
- Specialized for weak (witness) verification

Picus (https://github.com/chyanju/Picus)
- Picus is based on symbolic execution
- Supports customized specifications/queries besides weak (witness) uniqueness property
- Automated verification, less manual efforts required, incorporates optimizations from existing solvers

Problems & Existing Challenges
- Scalability: Difficult Constraints
- Coverage: Unsupported Cases
- Extensibility: New Emerging Language Interfaces
- ...
Potential Approaches for ZK Circuit Verification

Abstract Interpretation with Interval Analysis
- Obtain constraint annotations from user or static analysis

- **y = x;**
  - z = x - y;

  Constraint

  **Interval Analysis**

  - **y = x;**
    - z = x - y;

  Partially Annotated Constraint

  **Abstract Interpretation**

  - **x ∈ [1,3]**

  Annotated Constraints

  **Optimize**

  Solver

Unified Intermediate Representation for ZK Constraint Verification (Domain-Specific IR)
- CirC
- Vamp IR
- ...

Prime Field Theory for Existing Solvers
- Based on Gröbner bases solvers
- Based on Integer theory with annotated range intervals
- ...

Potential Approaches for ZK Circuit Verification
Plans & Next Steps

Core Library Circuits
- Core circomlib
- BigInt Arithmetic
- Elliptic Curve Arithmetic
- circom-ecdsa / circom-pairing

Application Circuits
- Application Circuit Benchmarks
- Constraint Annotation (Manual / Automated Analysis)
- Incorporation of Verified Core Library into Picus
- Abstract Interpretation for Uniqueness Analysis
THANKS