Introduction

Security Validation
- Validating the security of hardware designs is important.
- Developing a comprehensive set of security properties is challenging.

Goal
Reduce the manual effort for developing security properties.

Contribution
- A systematic approach for security property translation.
- Transys: a tool to translate security properties across hardware designs.

Background

Hardware Security Properties
- Restricted Temporal Logic
  - G(A → B), G means globally, may contain X (next) operator.
  - Example: AES round keys should be derived from the cipher key correctly.
    - (state == 4) → (next_key_reg[31:0] == next_key_reg[63:32] ⊕ last_key_reg[31:0])
- Gate Level Information Flow Tracking
  - Variables are tagged to track information flow is allowed to flow.
  - Example: The key is safe to flow to the ciphertext.
    - set key[0] := high; assert cipher[0] == high

Transys Design

Overview
- Gist: do the translation in three phases.
- Inputs: RTL implementations and a set of security properties.
- Manual review is still required.

Variable Mapping

- Goal: Find appropriate counterpart.
- Matching Windows
  - Modules in HDL
- Extracting Features from AST, PDG
  - Statistical
    - Variable Type (Input, Output, Wire, Reg)
    - No. of Blocking Assignments
    - No. of Non-Blocking Assignments
    - No. of Assignments
    - No. of Branch Conditions
    - No. of Always Block Conditions
  - Semantic
    - Variable Names
  - Structural
    - Dependence Graph Height
    - No. of Operators
    - Centroid

Structural Transformation

- Goal: Adjust arithmetic expressions.
- Observation
  - If in D1, the property variables are related to each other, in D2 the correlation among the variables are often explicitly stated in the code.
- Approach
  - Pick two most high-confidence variables
  - Learn the arithmetic expression between the variables from PDG.
- Timing Issue
  - For every nonblocking assignment, we add a X (next) to the property.

Constraint Refinement

- Goal: Make the property P valid in $\Phi_D$.
  - $\Phi_D \land \neg P$ is UNSAT.
  - Only consider Case 1 and 4.

Constraint: $\neg \alpha \land \neg x < \text{UNSAT}$.

Performance

Execution Time

<table>
<thead>
<tr>
<th>Design</th>
<th>Average Trans. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>28.8s</td>
</tr>
<tr>
<td>RSA</td>
<td>0.46s</td>
</tr>
<tr>
<td>CPU</td>
<td>189s</td>
</tr>
<tr>
<td>Average</td>
<td>70%</td>
</tr>
</tbody>
</table>

Transl. Examples

- Origin: (state == 4) → (next_key_reg[31:0] == next_key_reg[63:32] ⊕ last_key_reg[31:0])
- Pass Translation Results
  - VM: [key_exp.pstate==4]→[key_exp.key_in[31:0]==key_exp.key_in[63:32]]
  - ST: [key_exp.pstate==4]→[key_exp.key_in==256]
  - CR: [key_exp.key_start==128]→[key_exp.key_round==256]

Translation Results

- Experiment Setup
  - We collect 27 security properties in temporal logic and 9 in information flow tracking.
  - We evaluate Transys on 38 AES, 3 RSA, and 5 RISC processor designs.

- Designs | Total Transl. | Total Succ. | Rate |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>360</td>
<td>336</td>
<td>93%</td>
</tr>
<tr>
<td>AES w/ Trojans</td>
<td>400</td>
<td>400</td>
<td>100%</td>
</tr>
<tr>
<td>RSA</td>
<td>18</td>
<td>18</td>
<td>100%</td>
</tr>
<tr>
<td>CPU</td>
<td>46</td>
<td>39</td>
<td>85%</td>
</tr>
<tr>
<td>Total</td>
<td>824</td>
<td>793</td>
<td>96%</td>
</tr>
</tbody>
</table>