Transys: Leveraging Common Security Properties Across Hardware Designs

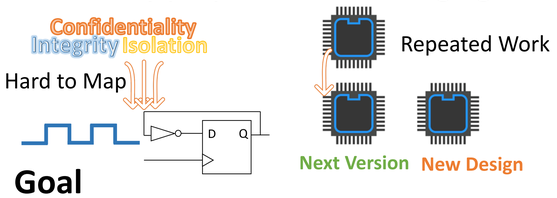
Rui Zhang, Cynthia Sturton (IEEE Symposium on Security and Privacy, S&P'20)

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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

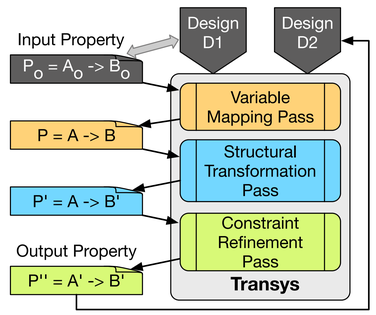
Hardware Security Properties

- **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_i[31:0])
- **Gate Level Information Flow Tracking**
  - Variables are tagged to track how information 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**

Type	Feature
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

- **Matching Variables**
  - $d_{stat}(p,q), d_{struct}(p,q) = \sqrt{(q_1-p_1)^2+\cdots+(q_n-p_n)^2}$
  - $d_{seman} = 1 - \frac{2\times|pairs(s_1)\cap pairs(s_2)|}{|pairs(s_1)|+|pairs(s_2)|}$
  - Distance between variables: $d(v_1,v_2) = \alpha d_{seman} + \beta d_{stat} + \gamma d_{struct}$
  - We choose α to be the largest and β to be the smallest.

Structural Transformation

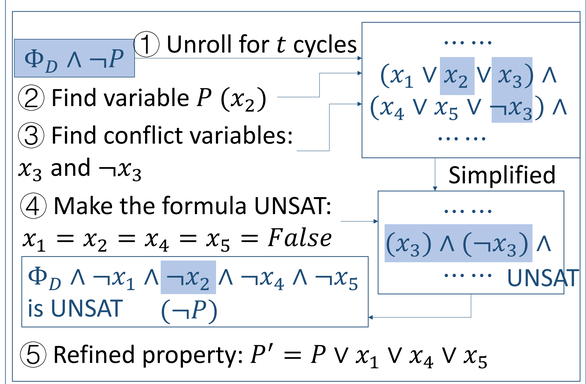
- **Goal:** Adjust arithmetic expressions.
- **Observation**
  - If in D1, the property variables are related to each other, in D2 the correlation among the variables are often explicitly stated in the code.
- **Approach**
  - Pick two most high-confidence variables
  - Learn the arithmetic expression 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 Φ_D.
  - $\Phi_D \wedge \neg P$ is UNSAT.
  - Only consider Case 1 and 4.

No.	Orig.	New Format	Simplified
1	A -> B	A ∧ C -> B	A ∧ C -> B
2		A ∨ C -> B	(A -> B) ∧ (C -> B)
3		A -> B ∧ D	(A -> B) ∧ (D -> B)
4		A -> B ∨ D	A ∧ ¬D -> B

  - Find a sequence of conjuncts such that $\Phi_D \wedge \neg P \wedge A_1 \wedge \cdots \wedge A_n$ is UNSAT, but $A_1 \wedge \cdots \wedge A_n$ is SAT.
- **Approach**
  - Fact: $x \wedge \neg x$ is UNSAT.



- Timing Issue: Consider one time step at a time.

Evaluation

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.

Translation Results

Designs	Total Transl.	Total Succ.	Rate
AES	360	336	93%
AES w/ Trojans	400	400	100%
RSA	18	18	100%
CPU	46	39	85%
Total	824	793	96%

Semantic Equivalence

Performance

Designs	Average Transl. Time
AES	28.8s
RSA	0.46s
CPU	189s
Average	70s

Transl. Examples

Orig: (state == 4) → (next_key_reg[31:0] == next_key_reg[63:32] ⊕ last_key_i[31:0])

Pass	Translation Results
VM	(key_exp.pstate==4)→(key_exp.key_in[31:0]== key_exp.key_in[63:32]⊕key_exp.key_in[31:0])
ST	(key_exp.pstate==4) → (key_exp.wr_data==key_exp.key_in[255:192])
	(key_exp.pstate==4) → (key_exp.wr_data==key_exp.key_in[191:128])
CR	(key_exp.key_start==1)&(key_exp.round[1:0]==2'b01)→#1 (key_exp.wr_data==prev(key_exp.key_in[255:192]))|(key_exp.wr3==0)
	(key_exp.key_start==0)&(key_exp.key_start_L==1) &(key_exp.round[1:0]==2'b01)→#1 (key_exp.wr_data==prev(key_exp.key_in[191:128]))|(key_exp.wr3==0)