TroPUF: Evaluating Hardware Trojan Insertion in Delay-Based Physical Unclonable Functions

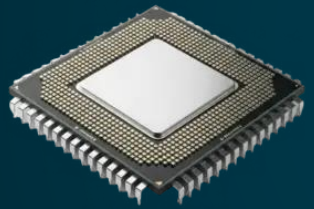
Marissa Marcarelli & Amin Rezaei

REFRESHER

MOTIVATION

We trust our devices to be secure.

We assume that the security primitives within them ensure protection and reliability.


What if these same primitives can be weaponized against the very systems they are meant to protect?

PRELIMINARY KNOWLEDGE

Physical Unclonable Functions (PUFs)

- Hardware security primitive
- Uses manufacturing variation
- Challenge → Response behavior (CRPs)
- Delay-based PUFs use signal timing imbalance
- Key properties:
  - Uniqueness
  - Reliability
  - Uniformity
  - Randomness

Hardware Trojans

- Malicious inserted hardware logic
- Consists of:
  - Trigger
  - Payload
- Remains dormant during normal operation
- Designed for stealth and rare activation
- Can alter system functionality or leak sensitive information

THE PROBLEM

What happens when a Trojan is inserted into the security primitive itself?

- Hardware Trojans are traditionally detected within deterministic, predictable circuits
- PUFs break all assumptions.
  - Intentionally random, noisy, and unique by design
- Inherent nontedeterminism could make PUFs an ideal carrier for malicious intent
- Malicious behavior may be masked or indistinguishable from normal PUF

PUFs themselves are assumed to be trusted.

- Trojan behavior within the PUF primitive itself remains unstudied.
- Trojan insertion into a PUF exploits the very properties that make PUFs useful.
- The very primitive designed to ensure system security may also be what compromises it.

OBJECTIVES

Establish the feasibility of embedding hardware Trojans within delay-based PUFs under realistic conditions.

- Before detection within PUFs can be studied, it must first be proven that stealthy Trojan insertion is possible.
- Evaluate whether hardware Trojans can be embedded within delay-based PUFs and successfully execute payloads.
- Determine if Trojans can maintain stealthy, dormant behavior while preserving normal PUF functionality.
- Assess whether PUF nondeterminism can conceal Trojan presence.

RELATED WORKS

Delay-Based PUFs

- Arbiter, XOR, Feedforward, Interpose, and Cyclic PUF architectures
- Prior work focused on modeling resistance and statistical metrics

Modeling Attacks

- Delay-based PUFs shown to be learnable under supervised ML approaches
- Increasing architectural complexity does not guarantee security

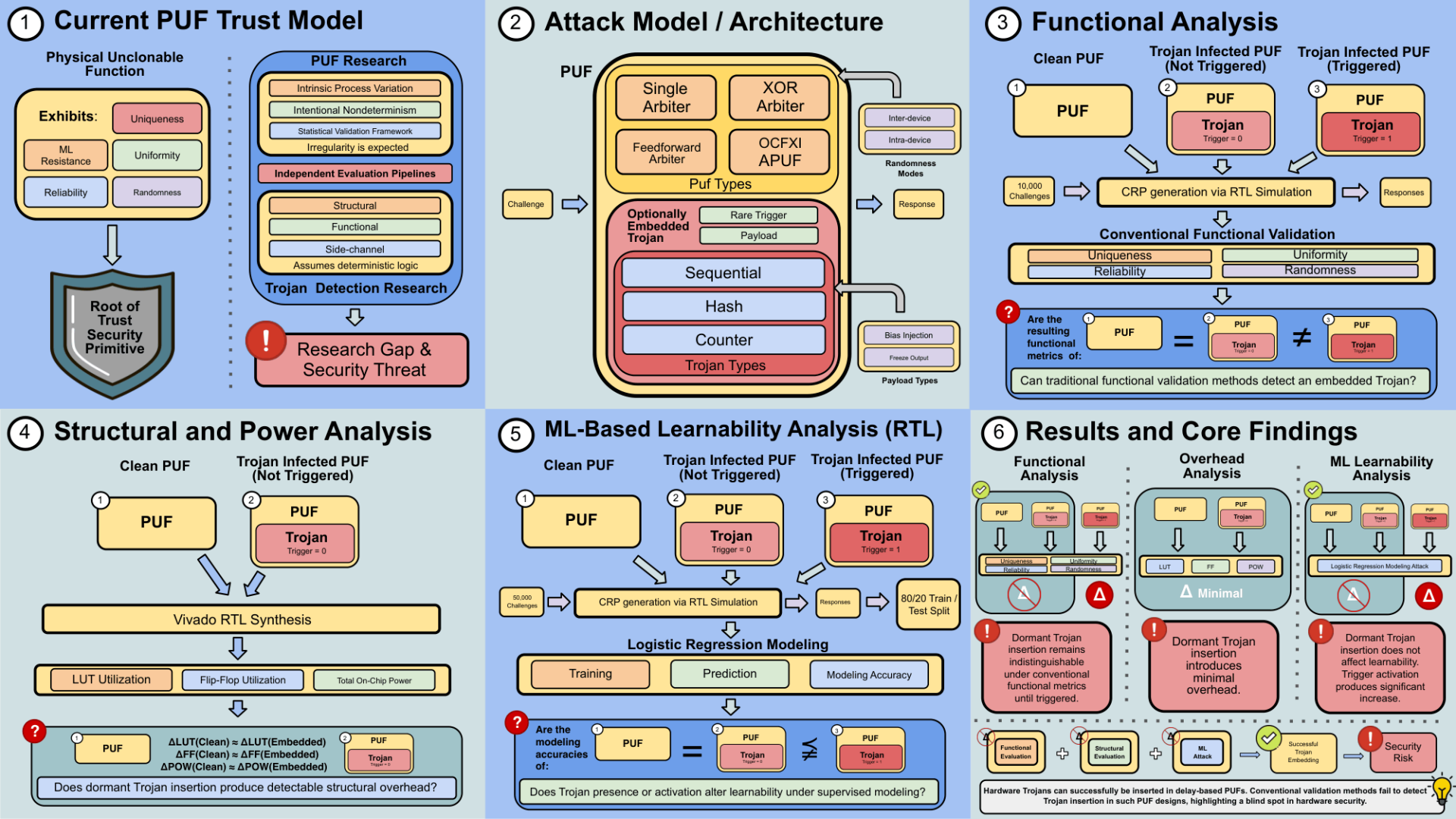
Hardware Trojans

- Extensive work on stealthy triggers, low overhead insertion, and rare activation
- Conventional detection relies on deterministic assumptions

Research Gap

- Very limited work evaluates Trojans embedded directly inside PUFs
- Existing validation methods do not account for inherent nondeterminism

CONTRIBUTIONS


1 Current PUF Trust Model
Physical Unclonable Function
Exhibits:
Uniqueness
ML Resistance
Uniformity
Reliability
Randomness
Root of Trust Security Primitive
PUF Research
Intrinsic Process Variation
Intentional Nondeterminism
Statistical Validation Framework
Irregularity is expected
Independent Evaluation Pipelines
Structural
Functional
Side-channel
Assumes deterministic logic
Trojan Detection Research
Research Gap & Security Threat
2 Attack Model / Architecture
PUF
Single Arbiter
XOR Arbiter
Feedforward Arbiter
OCFXI APUF
Puf Types
Inter-device
Intra-device
Randomness Modes
Challenge
Response
Optionally Embedded Trojan
Rare Trigger
Payload
Sequential
Hash
Counter
Trojan Types
Bias Injection
Freeze Output
Payload Types
3 Functional Analysis
Clean PUF
Trojan Infected PUF (Not Triggered)
Trojan Infected PUF (Triggered)
PUF
Trojan
Trigger = 0
Trigger = 1
10,000 Challenges
CRP generation via RTL Simulation
Responses
Conventional Functional Validation
Uniqueness
Uniformity
Reliability
Randomness
Are the resulting functional metrics of:
Can traditional functional validation methods detect an embedded Trojan?
4 Structural and Power Analysis
Clean PUF
Trojan Infected PUF (Not Triggered)
Vivado RTL Synthesis
LUT Utilization
Flip-Flop Utilization
Total On-Chip Power
ΔLUT(Clean) ≈ ΔLUT(Embedded)
ΔFF(Clean) ≈ ΔFF(Embedded)
ΔPOW(Clean) ≈ ΔPOW(Embedded)
Does dormant Trojan insertion produce detectable structural overhead?
5 ML-Based Learnability Analysis (RTL)
Clean PUF
Trojan Infected PUF (Not Triggered)
Trojan Infected PUF (Triggered)
50,000 Challenges
CRP generation via RTL Simulation
Responses
80/20 Train / Test Split
Logistic Regression Modeling
Training
Prediction
Modeling Accuracy
Are the modeling accuracies of:
Does Trojan presence or activation alter learnability under supervised modeling?
6 Results and Core Findings
Functional Analysis
Overhead Analysis
ML Learnability Analysis
Uniqueness
Uniformity
Reliability
Randomness
LUT
FF
POW
Δ Minimal
Logistic Regression Modeling Attack
Dormant Trojan insertion remains indistinguishable under conventional functional metrics until triggered.
Dormant Trojan insertion introduces minimal overhead.
Dormant Trojan insertion does not affect learnability. Trigger activation produces significant increase.
Functional Evaluation
Structural Evaluation
ML Attack
Successful Trojan Embedding
Security Risk
Hardware Trojans can successfully be inserted in delay-based PUFs. Conventional validation methods fail to detect Trojan insertion in such PUF designs, highlighting a blind spot in hardware security.

ARCHITECTURE

PUFs

Single Arbiter PUF

- Baseline delay-based architecture using symmetric racing paths
- Response determined by relative signal arrival time
- Used as foundational reference implementation

Feedforward Arbiter PUF

- Intermediate arbiter outputs are fed back into later mux stages
- Earlier race results dynamically alter downstream path selection
- Creates recursive internal dependencies within the delay chain

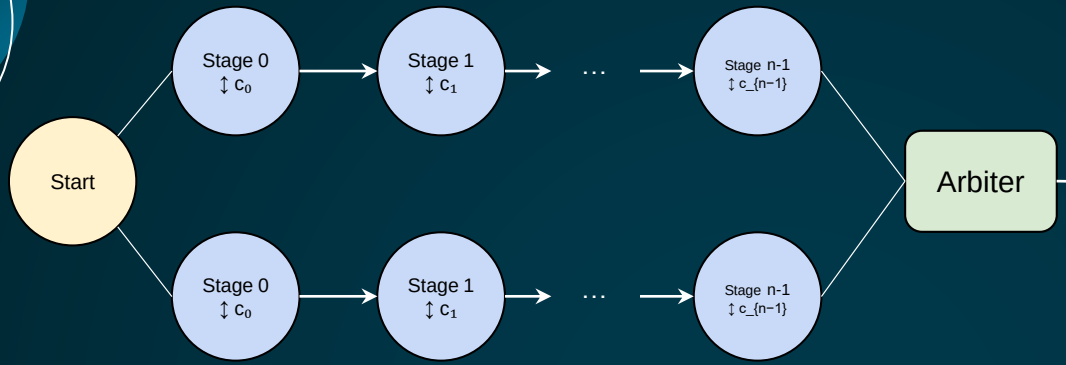
XOR Arbiter PUF

- Combines multiple Arbiter PUF outputs using XOR logic
- Introduces additional nonlinearity
- Known to increase resistance to conventional modeling attacks

OCFXI Arbiter PUF

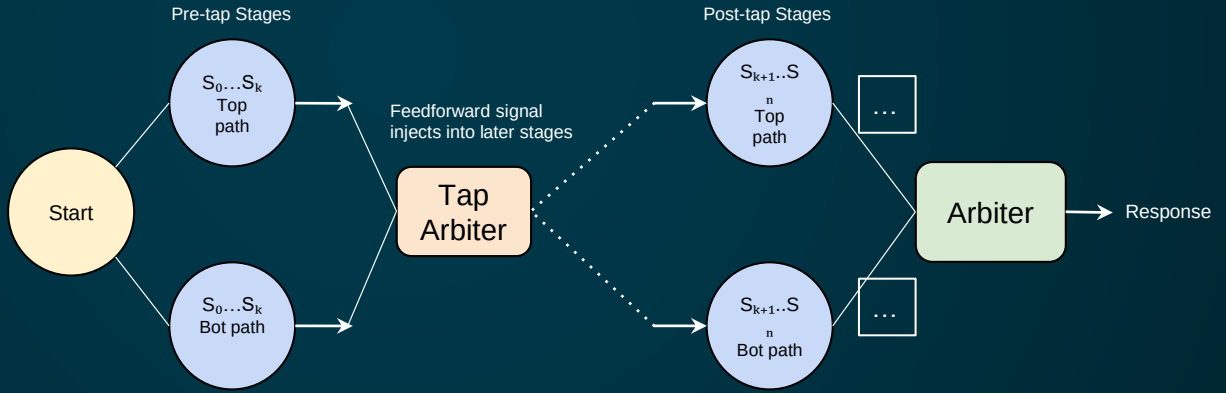
- Permuted challenge evaluated through upper ffw arbiter bank
- Intermediate responses injected into lower XOR arbiter bank inputs
- Upper bank behavior dynamically modifies lower bank delay paths
- Creates hierarchical nonlinear CRP behavior

SINGLE ARBITER



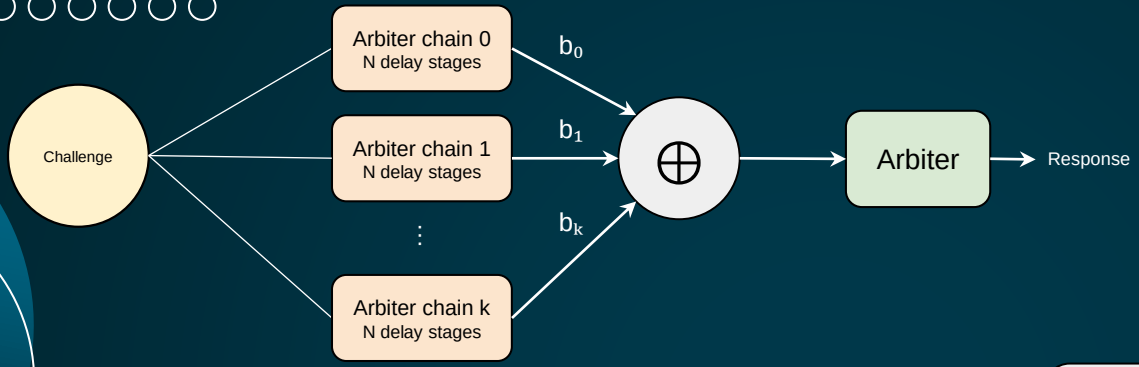
manufacturing variation → asymmetric delays → arbiter resolves race

FEED-FORWARD APUF



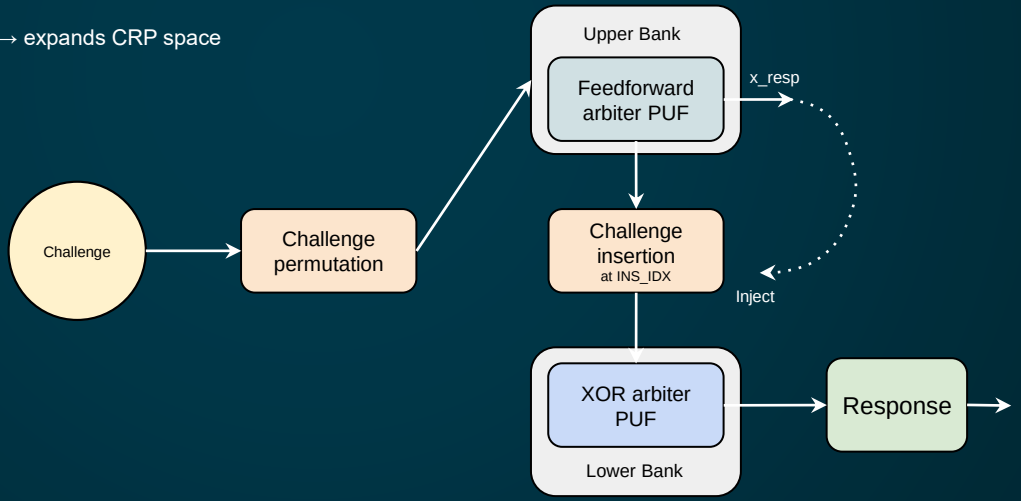
feedforward breaks linear separability → increases modeling resistance

XOR APUF



k independent chains & nonlinear XOR → expands CRP space

OCFXI APUF



upper bank output modifies lower bank input → hierarchical nonlinearity

TROJANS

Sequential Trigger

- Finite-state machine monitors ordered challenge sequences
- Progress resets if incorrect challenge appears
- Activates only after full sequence completion

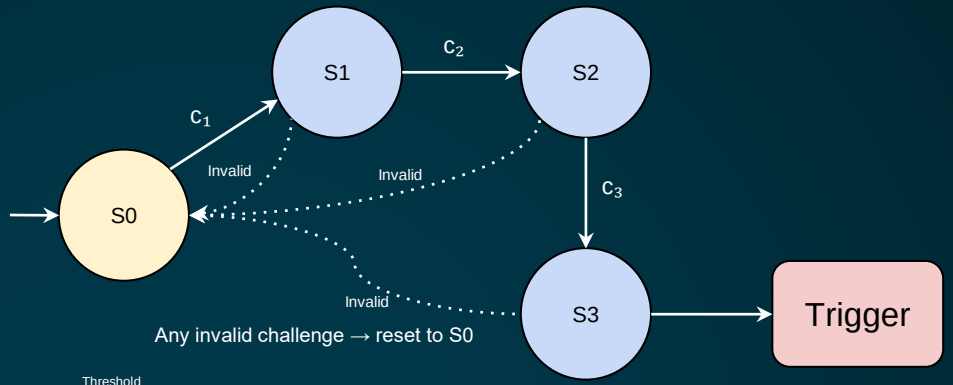
Counter Trigger

- Internal counter increments on each PUF evaluation
- Trigger activates only after threshold is reached
- Models long-latency dormant behavior

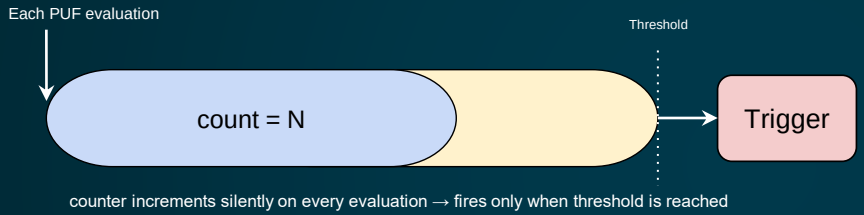
Hash Trigger

- Lightweight hash computed over challenge vector
- Trigger activates only when hash matches hidden target
- Obfuscates activation condition from direct input patterns

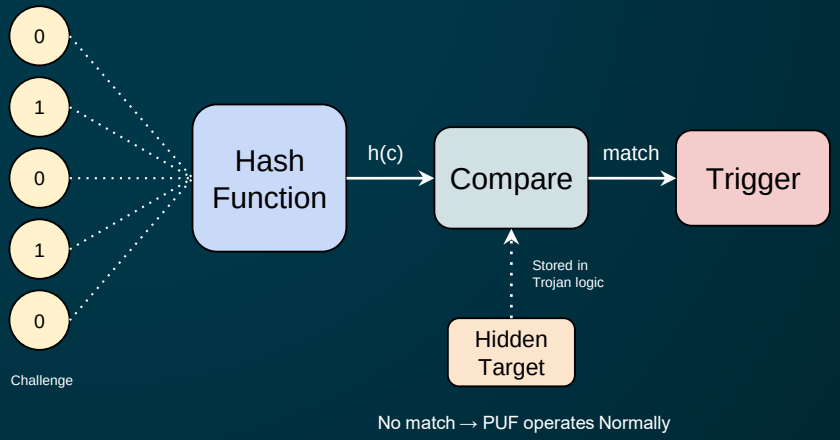
SEQUENTIAL



COUNTER



HASH



PAYLOADS

Bias Injection

- Triggered payload forces responses toward a target logic value
- PUF continues operating, but output distribution becomes skewed
- Degrades statistical properties such as uniformity and randomness

Freeze Output

- First valid response after activation is stored internally
- Stored value reused for all future evaluations
- Collapses CRP variability while preserving internal activity

Payloads remain inactive during standard operation and only modify behavior after trigger activation.

FRAMEWORK INFRASTRUCTURE

Centralized Testbench

- Controls challenge application and simulation flow
- Automates response collection and experiment logging
- Evaluates clean and Trojan-infected configurations under identical conditions

Parameterized Wrapper

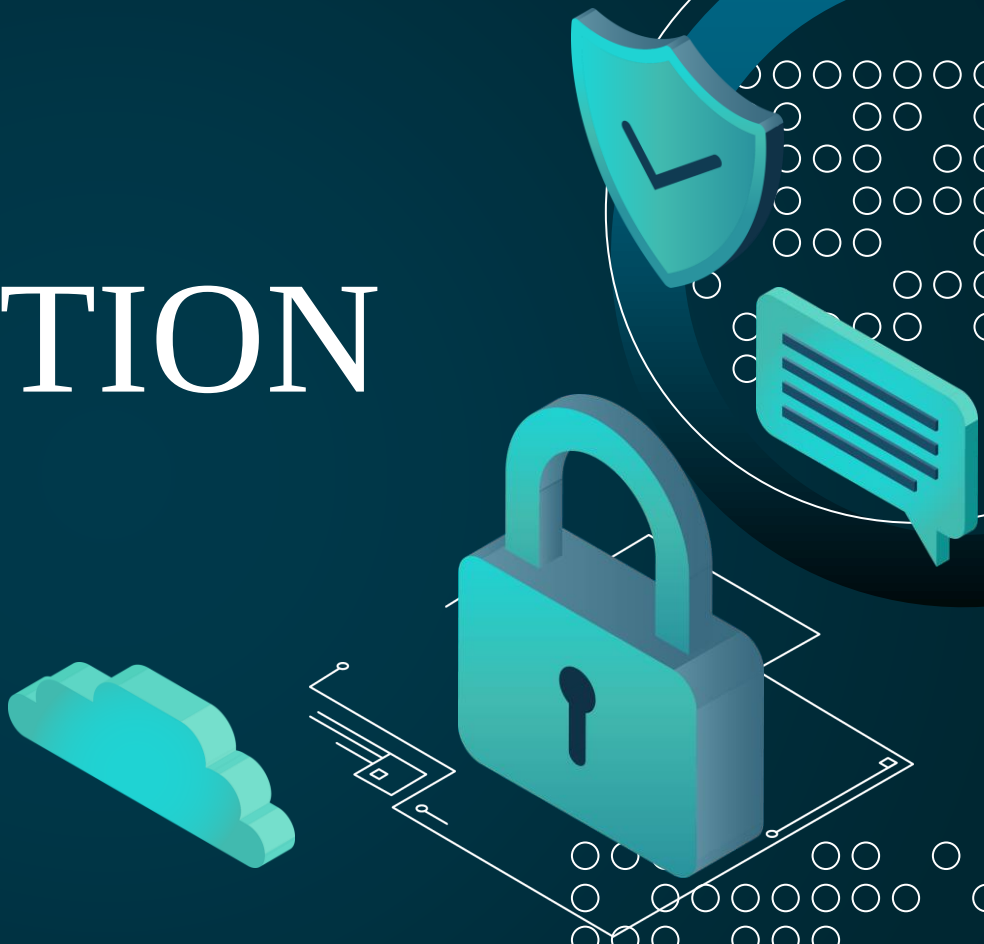
- Unified interface across all PUF architectures
- Configurable selection of PUF type, trigger type, payload, randomness mode
- Encapsulates Trojan and PUF integration logic

Shared Modules

- Reusable delay stages, arbiters, triggers, and payload components
- Maintains implementation consistency across experiments
- Supports scalable architecture expansion and reproducibility

Modular design enables controlled large-scale evaluation across multiple architecture and attack combinations.

EVALUATION



EVALUATION METRICS

Functional Metrics

- Measured whether dormant Trojans altered the expected behavior of PUF characteristics
  - Uniqueness - variation in responses across different devices
  - Reliability - consistency of responses for the same device
  - Uniformity - balance of 0s and 1s across outputs
  - Randomness - statistical unpredictability of responses

Synthesis Analysis

- Assessed whether Trojan insertion produces detectable hardware level changes
  - Lookup Table Utilization (LUT) - combinational logic usage
  - Flip-Flop Utilization - sequential storage elements used
  - Power Overhead - additional power consumption introduced

ML Modeling Attack

- Attempt to model PUF behavior from CRPs
- Evaluated whether Trojan presence affects PUF modeling resistance
- Assess whether embedded Trojans are detectable using machine learning
- Applied logistic regression on 50,000 CRPs

EXPERIMENTAL DESIGN

Tools

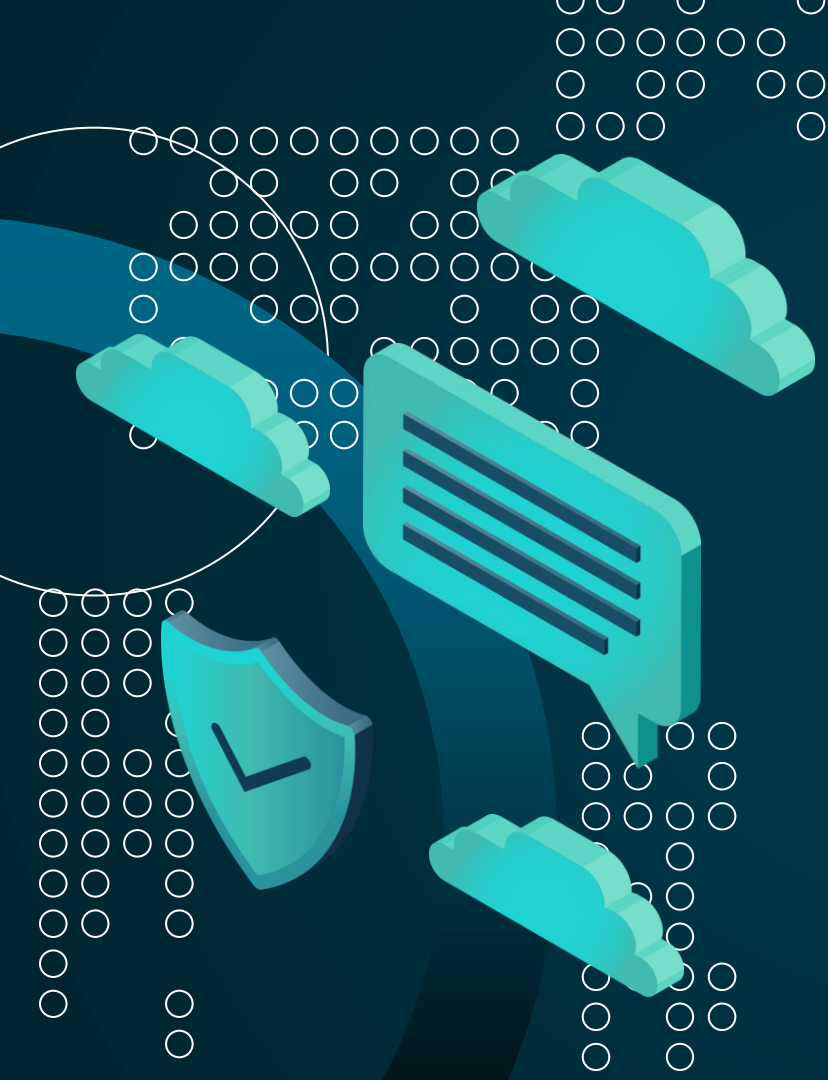
- Verilog RTL simulation framework
- Xilinx Vivado for synthesis and power analysis
- Python automation for CRP generation, logging, and metric evaluation
- Logistic regression used for ML learnability analysis

Randomness Modeling

- Intra-device - consistency of one PUF instance across repeated evaluations
- Inter-device - distinctiveness between separate PUF instances
- Optional randomness modes applied across all architectures

Scalability

- Parameterized compile-time configuration
- Automated challenge generation and dataset creation
- Unified evaluation flow across all PUF/Trojan combinations
- Supports large scale reproducible experimentation



RESULTS

FUNCTIONAL METRICS

Functional Evaluation

- Clean and dormant Trojan designs were statistically indistinguishable across all metrics and PUF/Trojan types
  - Near ideal values (~50% uniqueness, 100% reliability, ~50% uniformity, ~100% randomness)
- PUF functionality unaltered by Trojan Presence
- Metric degradation appeared only after activation, and was immediate and severe across all architectures

UNIQUENESS VALUES ACROSS PUF IMPLEMENTATIONS (%)

Configuration	SA	FF	XOR	OCFXI
Clean v. Clean	48.9	49.6	50.2	49.8
Clean v. Sequential Trojan (Dormant)	48.9	49.6	50.2	49.9
Clean v. Sequential Trojan (Triggered)	1.2	0.9	0.6	0.4
Clean v. Hash Trojan (Dormant)	48.9	49.6	50.2	49.9
Clean v. Hash Trojan (Triggered)	1.1	0.8	0.5	0.3
Clean v. Counter Trojan (Dormant)	48.9	49.6	50.2	49.9
Clean v. Counter Trojan (Triggered)	1.3	1.0	0.7	0.5

RELIABILITY VALUES ACROSS PUF IMPLEMENTATIONS (%)

Configuration	SA	FF	XOR	OCFXI
Clean v. Clean	100	100	100	100
Clean v. Sequential Trojan (Dormant)	100	100	100	100
Clean v. Sequential Trojan (Triggered)	49.8	50.2	49.5	50.1
Clean v. Hash Trojan (Dormant)	100	100	100	100
Clean v. Hash Trojan (Triggered)	50.3	49.7	50.0	49.6
Clean v. Counter Trojan (Dormant)	100	100	100	100
Clean v. Counter Trojan (Triggered)	49.9	50.4	49.6	50.2

UNIFORMITY VALUES ACROSS PUF IMPLEMENTATIONS (%)

Configuration	SA	FF	XOR	OCFXI
Clean	48.7	49.4	50.1	49.8
Sequential Trojan (Dormant)	48.7	49.4	50.1	49.8
Sequential Trojan (Triggered)	99.2	99.5	99.7	99.9
Hash Trojan (Dormant)	48.7	49.4	50.1	49.8
Hash Trojan (Triggered)	99.1	99.6	99.8	99.9
Counter Trojan (Dormant)	48.7	49.4	50.1	49.8
Counter Trojan (Triggered)	99.3	99.4	99.7	99.8

RANDOMNESS VALUES ACROSS PUF IMPLEMENTATIONS (%)

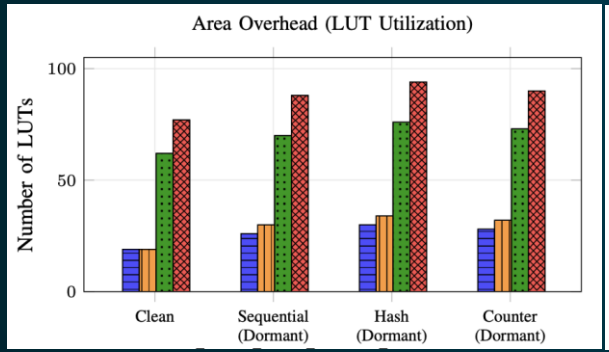
Configuration	SA	FF	XOR	OCFXI
Clean	99.89	99.99	99.98	99.99
Sequential Trojan (Dormant)	99.89	99.99	99.98	99.99
Sequential Trojan (Triggered)	2.56	1.98	0.87	0.15
Hash Trojan (Dormant)	99.99	99.99	99.98	99.99
Hash Trojan (Triggered)	1.47	0.92	0.41	0.12
Counter Trojan (Dormant)	99.99	99.99	99.98	99.99
Counter Trojan (Triggered)	1.43	1.01	0.76	0.20

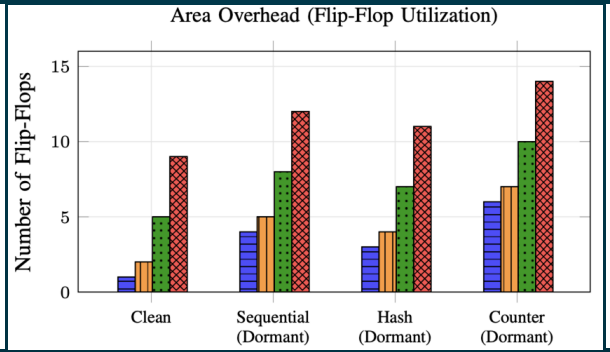
22

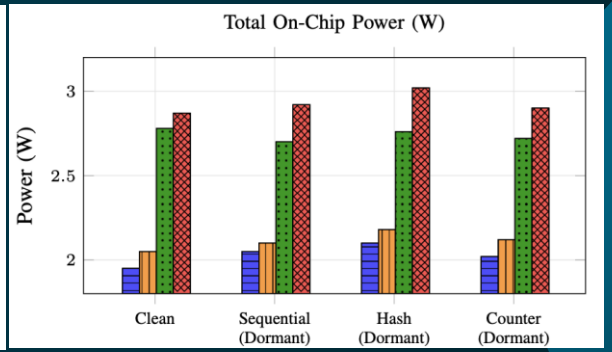
SYNTHESIS ANALYSIS

Synthesis Analysis

- Dormant Trojan insertion introduced slight but consistent increases in LUT and Flip-Flop utilization across all architectures
- Power analysis showed no consistent or highly detectable signature attributable to dormant Trojan presence
- Dormant Trojan insertion produces no reliable structural detection signal







ML MODELING ATTACK

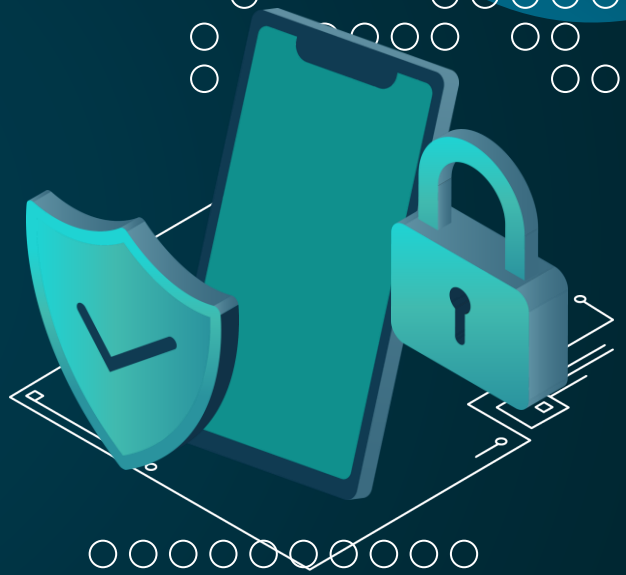
ML Modeling Attack

- All clean and dormant configurations around ~50% prediction accuracy, equivalent to random guessing
- Dormant Trojan presence had no observable effect on modeling accuracy
- Upon activation, accuracy increased to ~99.5–99.8% across all architectures, proving
- Trojan presence alters the modeling resistance of the PUFs after activation as expected, while remaining completely stealthy prior to triggering

ML Model Prediction Accuracy Across PUF Implementations (%)

Configuration	SA	FF	XOR	OCFXI
Clean	49.3	51.3	49.6	49.7
Sequential Trojan (Dormant)	49.3	51.3	49.6	49.7
Hash Trojan (Dormant)	49.3	51.3	49.6	49.7
Counter Trojan (Dormant)	49.3	51.3	49.6	49.7
Sequential Trojan (Triggered)	99.5	99.7	99.8	99.7
Hash Trojan (Triggered)	99.7	99.7	99.8	99.9
Counter Trojan (Triggered)	99.4	99.6	99.7	99.6

DISCUSSION



MAIN FINDINGS

Trojan-infected PUFs are indistinguishable from clean designs under conventional evaluation.

- Dormant Trojans do not alter PUF behavior
- Conventional validation fails to detect Trojan presence
- Trojan infected designs appear identical to clean implementations under traditional evaluation pipelines
- Detection only occurs after Trojan activation

PUF variability conceals malicious logic until activation.

WHY DO WE CARE

Security Implications

- Passing standard tests and appearing “correct” does not imply integrity or safety.
- Conventional PUF validation methods fail to detect injected Trojans prior to activation.
- A PUF can appear statistically ideal, structurally unchanged, and ML resistant, yet contain malicious logic.
- The properties that enable PUFs security are the same properties that enable Trojan concealment, in turn making them dangerous.

Research Impact

- Establishes a unified, scalable, and reproducible framework for evaluating Trojan insertion across multiple delay-based PUF architectures.
- Enables extension to additional PUF architectures, Trojan types, and attack models.
- Demonstrates that realistic PUF behavior, stealthy Trojans, and ML resistance can coexist simultaneously, highlighting a security risk.
- Serves as a foundation for developing PUF based Trojan detection and defense mechanisms.

THANK YOU!
QUESTIONS?

Source Code: