Interruptible Remote Attestation via Performance Counters

Davide Li Calsi
davide.li@mail.polimi.it

Supervisor: Prof. Vittorio Zaccaria

POLITECNICO
MILANO 1863

HONOURS PROGRAMME
HP-SR
in Information Technology

Agenda

- Remote Attestation (RA)
- RA and Interrupts
- Performance Counters for Malware Detection
- A new approach to interruptible RA
- Target System’s Architecture
- Experiment Design
- Experimental Results
- Conclusions

Remote Attestation

My message to companies that think they haven’t been attacked is: ‘You’re not looking hard enough.’

James Snook

Low-end MCUs and security

Low computational power

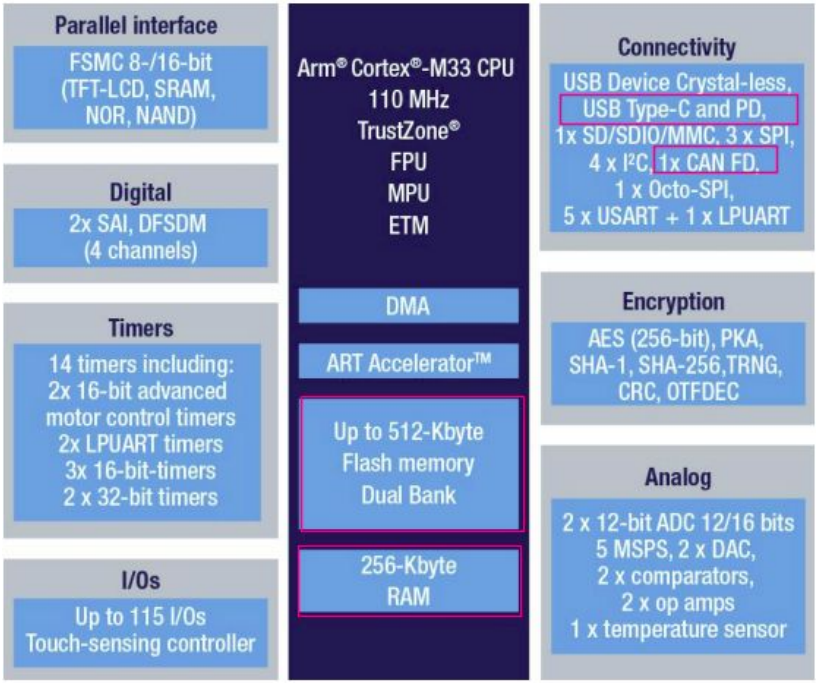
Hundreds of KB of RAM/FLASH

Little hardware protection.

Cheap and flexible.

Employed in several use-cases

ST-Nucleo L-552 board by STM32



Attacks*

- Access to sensors to collect sensitive data
  - Use microphones to spy on conversations
  - Use sensors to spot empty houses and rob them
- Control actuators to cause accidents
  - Smart ovens that caused fires
  - Taking over smart lights to cause epileptic seizures

*A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8742551

Remote Attestation*

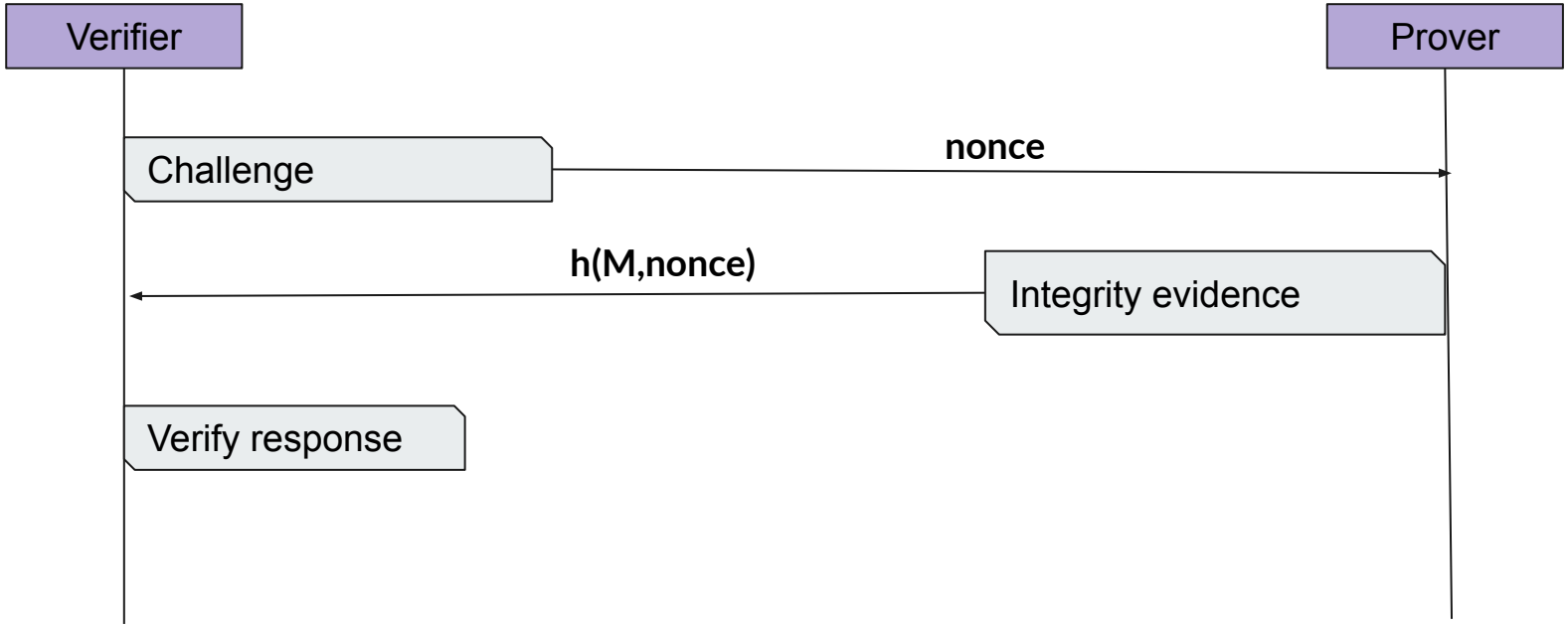
Remote Verifier attests the integrity of a **target Prover.**

Detect compromised devices.

Focus on **static attestation:** attest *Program Memory only*

*Remote Attestation: A Literature Review: https://arxiv.org/abs/2105.02466

Remote Attestation (2)

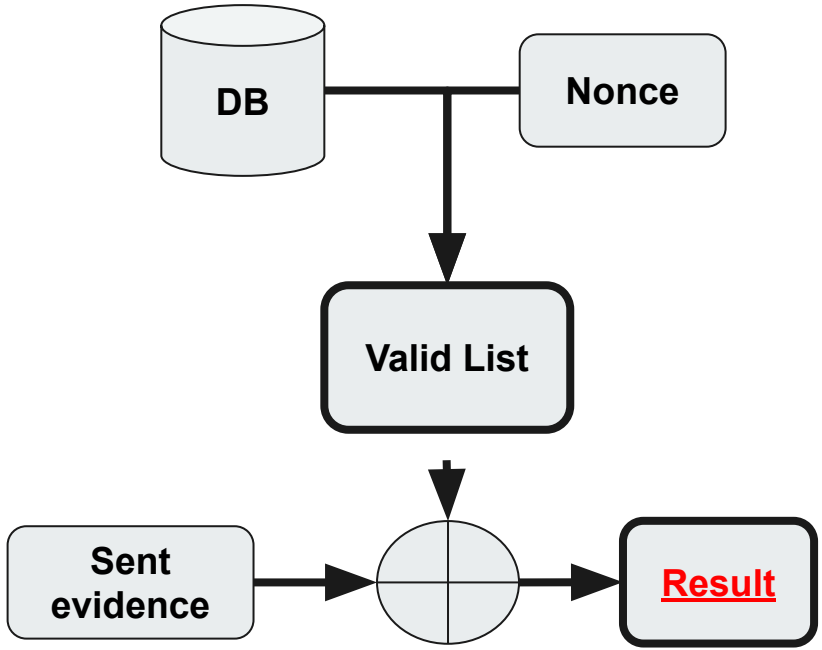


Verification

Static attestation: know benign configurations

Pre-compute benign evidence.

Compare with the received one.

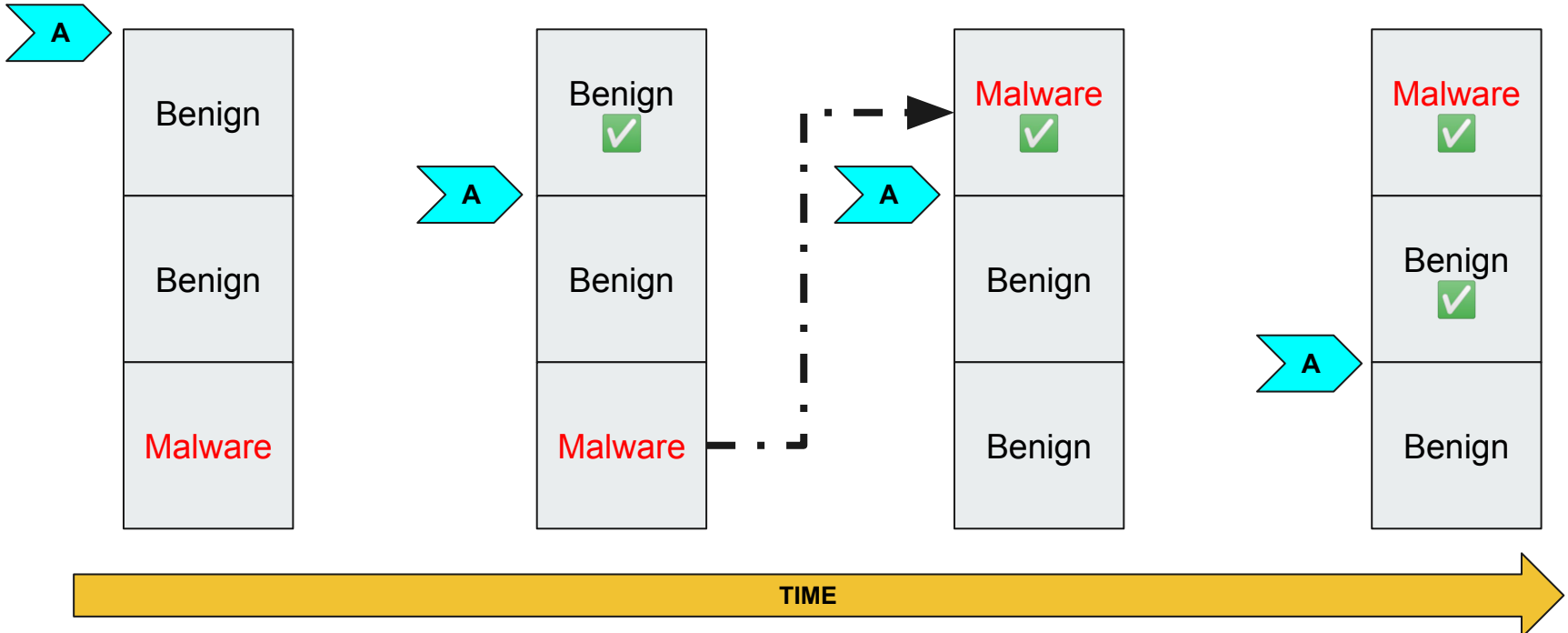


RA and Interrupts

I haven’t spoken to my wife in years. I didn’t want to interrupt her.

Rodney Dangerfield

Interrupts in RA - Relocation



Interrupts in RA - Disabled

Must disable interrupts to fight **roving malware**.

Self-relocating: erases itself and moves to a location that was already attested.

Transient malware: self-erases to later re-infect the system.

The main issue

Interrupts are a **key feature** , should not be disabled for too long.

Grant the system responsiveness.

Order of magnitude: **hundreds of ms** on average.

Time-critical cannot tolerate it.

Shuffled Measurements Against Roving Malware*

Attest in pseudo-random order

Probabilistic guarantees, depending on the attacker's knowledge (63%).

Repeat attestation multiple times in a row to increase probabilities.

Partial interruption: attestation of single blocks is still atomic.

*Remote attestation of IoT devices via SMARM: Shuffled measurements against roving malware: https://ieeexplore.ieee.org/document/8383885

Memory Locks*

Make memory read-only (temporarily)

Many implementations, depending on what you lock.

Based on costly system calls and MMU. Unfeasible for low end devices.

*Temporal Consistency of Integrity-Ensuring Computations and Applications to Embedded Systems Security: https://dl.acm.org/doi/10.1145/3196494.3196526

Performance Counters for Malware Detection

What we can control is our performance and execution, and that's what we are focusing on.

Bill Belichick

Which counters?

Natively present in many renown architectures.

ARM: **PMU** or **DWT**

Atmel: configurable **16-bits counters**

Count the occurrences of micro-architectural events:

- branches
- cache hits/misses
- CPI
- clock cycles spent doing something meaningful

Two phases

Offline phase: run some attacks, collect counters, train a classifier

Online phase: feed counters to the pre-trained classifier, classify them

Classify the result

Several models were evaluated*.

Decision Trees, SVM, K-NN…

Papers report accuracies between 60% and 90% (and above)

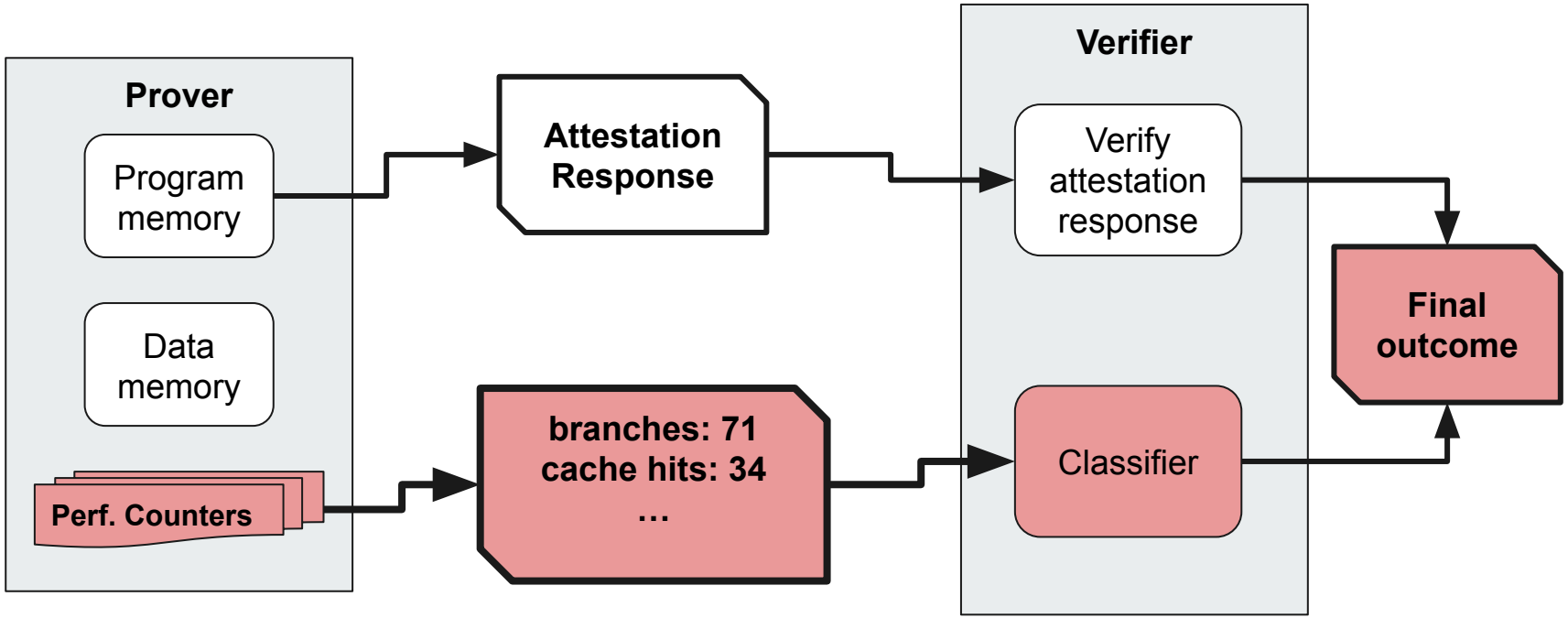
*ConFirm: Detecting firmware modifications in embedded systems using Hardware Performance Counters:https://ieeexplore.ieee.org/document/7372617
HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition: https://ieeexplore.ieee.org/document/6993402

A new approach to interruptible RA

Making progress on longstanding challenges requires a different lens and a new approach

Ayanna Presley

Summing-up


Prover
Program memory
Data memory
Perf. Counters
Attestation Response
branches: 71
cache hits: 34
…
Verifier
Verify attestation response
Classifier
Final outcome

Two phases (Again)

Offline phase: run several malicious relocations that interrupt the Attestation Routine. Obtain data and train a Binary Classifier.

Online phase: model classifies the measured counters. Determine if malware tried to escape detection.

Counters

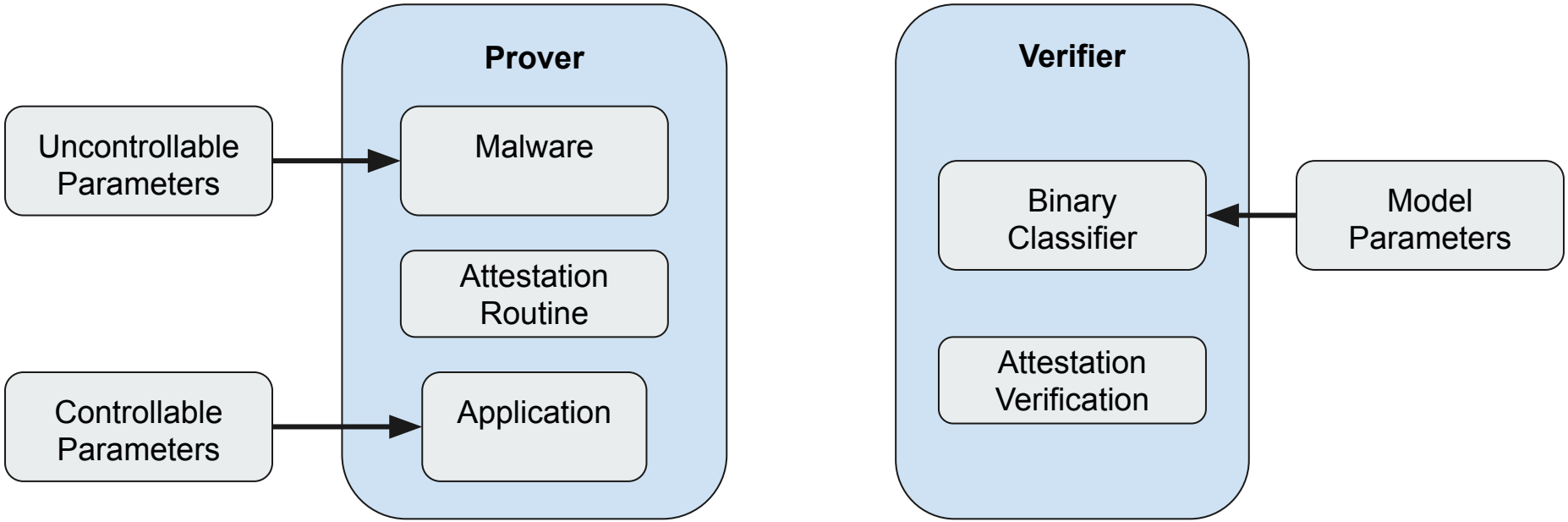
- **Architectural counters**
  - **Hardware-**managed
  - Count micro-architectural events
  - Literature agrees they are beneficial
- **Applicative counters**
  - **Software-**managed
  - Count high-level events
  - Controversy: overhead, protection, definition

Target System's Architecture

Each new situation requires a new architecture

Jean Nouvel

Overview

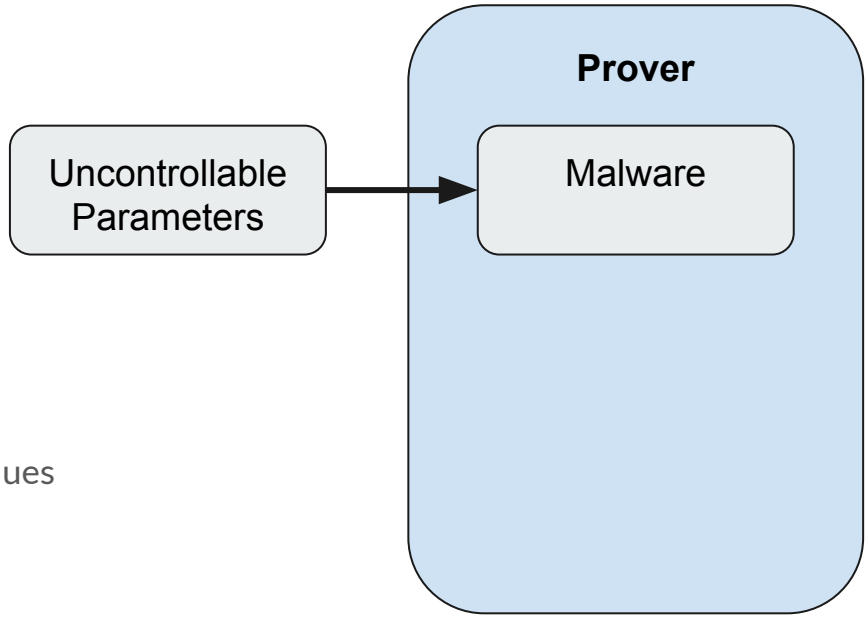

Prover
Uncontrollable Parameters
Malware
Attestation Routine
Controllable Parameters
Application
Verifier
Binary Classifier
Model Parameters
Attestation Verification

Uncontrollable Parameters

Parameters characterizing the attack type.

Attacks are unpredictable

1. **Malware Type**
    a. ***Self-relocating***
    b. ***Transient***
2. **Malware Size**
    a. Taken from a reasonable set of possible values

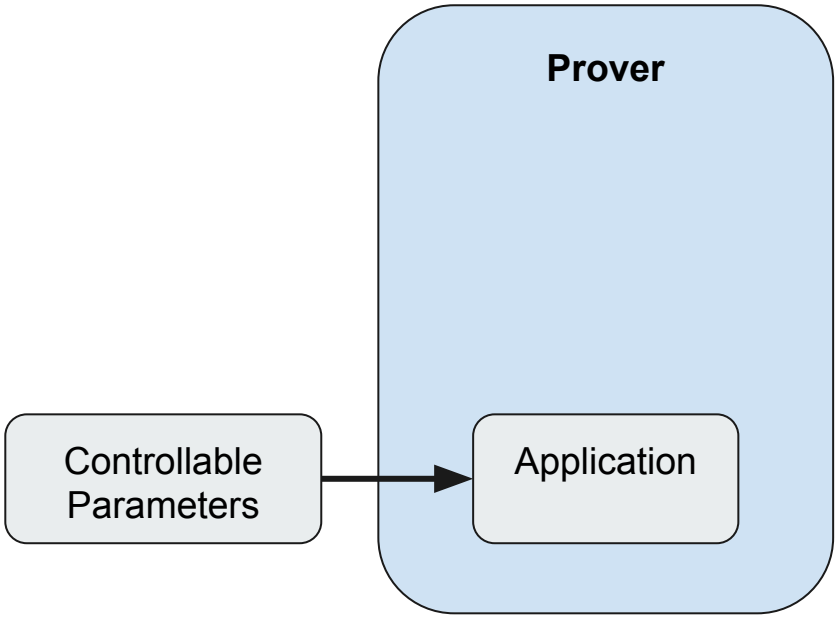


Controllable Parameters

Characterize the application running on the MCU.

Controllable because defined and controlled by the stakeholders.

1. **Entropy Level:** the degree of unpredictability of the application.
2. **Activity Level:** the intensity of the application's activity.



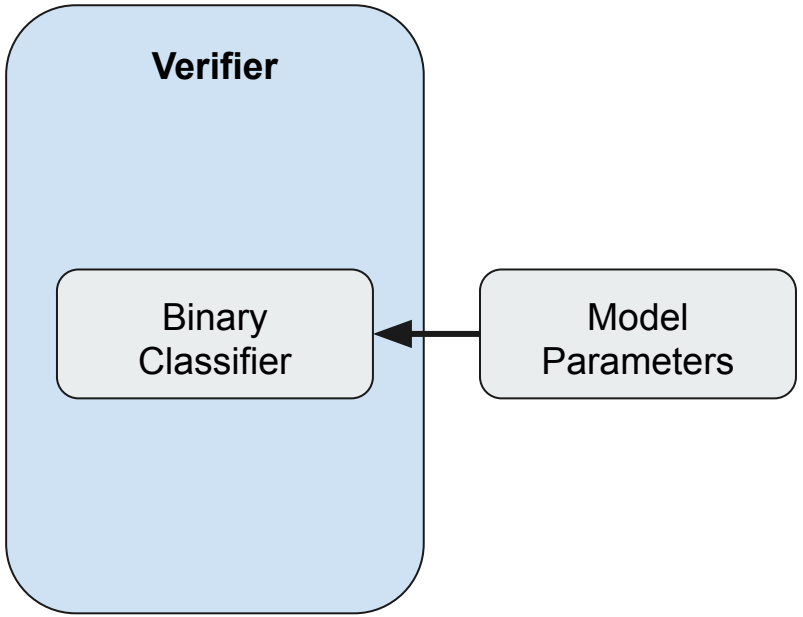
Model's Parameters

Several choices affect the Classifier:

- model
- preprocessing
- enhancements

Also under the stakeholders' control, but…

…we consider them separately because they are **Verifier-side** parameters.



Experiment Design

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.

Richard P. Feynman

Goals

G1: Detection Capabilities

Classifier should have satisfying:

- **Accuracy**
- **Precision**
- **Recall** (detect malware)
- **F1 score** (imbalanced dataset)

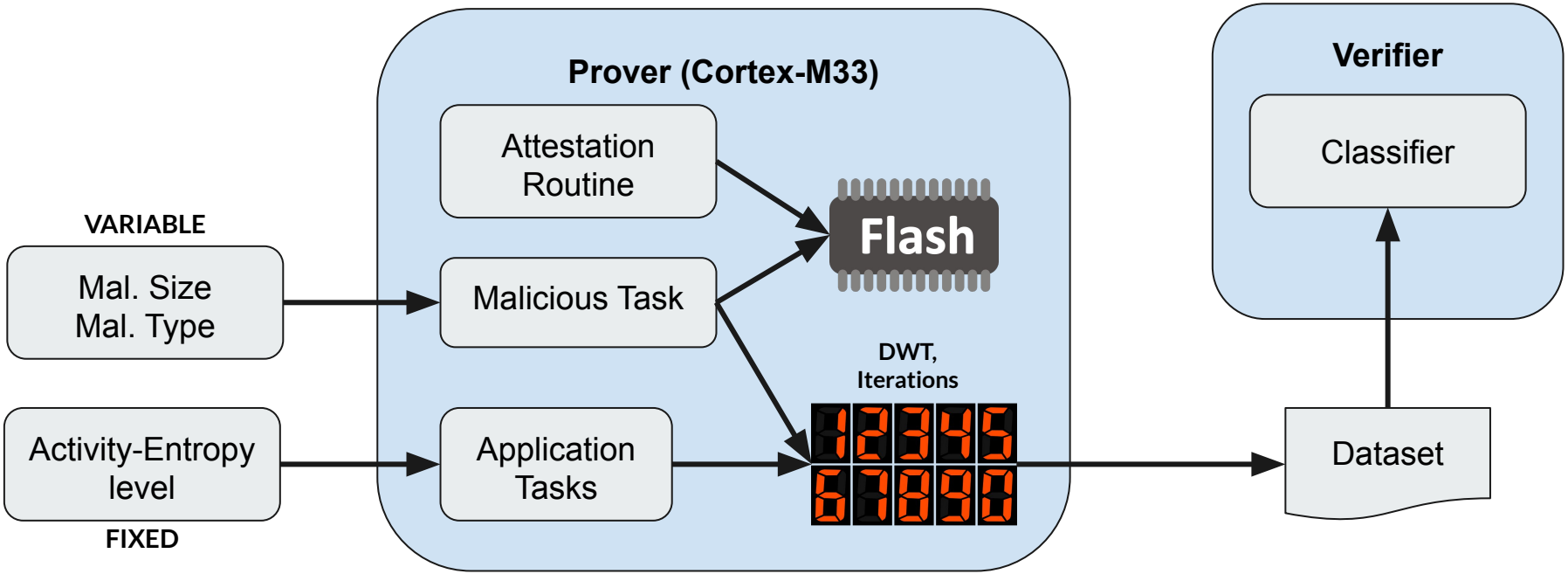
G2: Overhead

Low-power assumption, technique should be lightweight.

Questions to answer

1. Detection capability of **architectural counters**
2. Detection capability of **architectural counters** and **applicative counters**
3. Improve architectural counters?
4. Most relevant counters?
5. Overhead:
    a. hardware role
    b. application role

Prover/verifier setup



Varying (Un)Controllable Parameters

Malware type: only two values (self-relocating vs transient)

Size: low-power MCU host small malware samples.

Fixed a set of reasonable sizes (checking malware repositories)

Repeated **16 times** with different Activity-Entropy combinations…

… for a total of 16 datasets

Classifiers

Tested three Classifiers:
Logistic Regression, Decision Tree, Support Vector Machine

Using *scikit-learn* library for implementations

Each model trained and tested on every dataset.

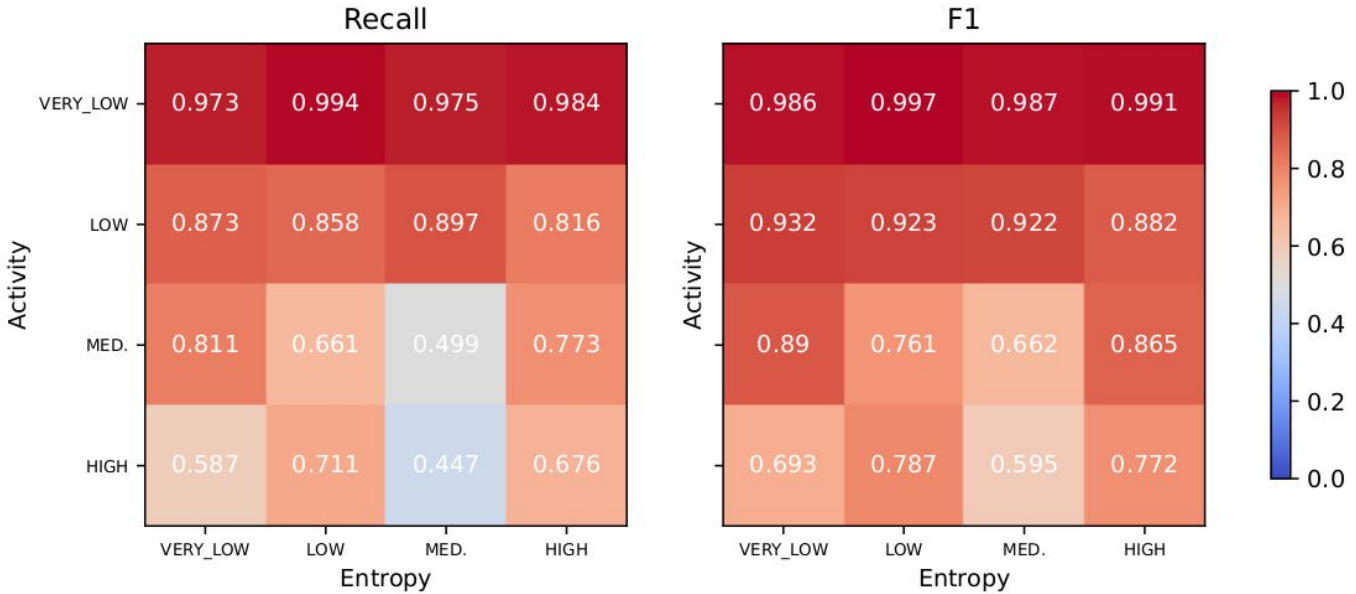
Results are 4x4 matrices (each square ≡ Activity-Entropy combination)

Experimental Results

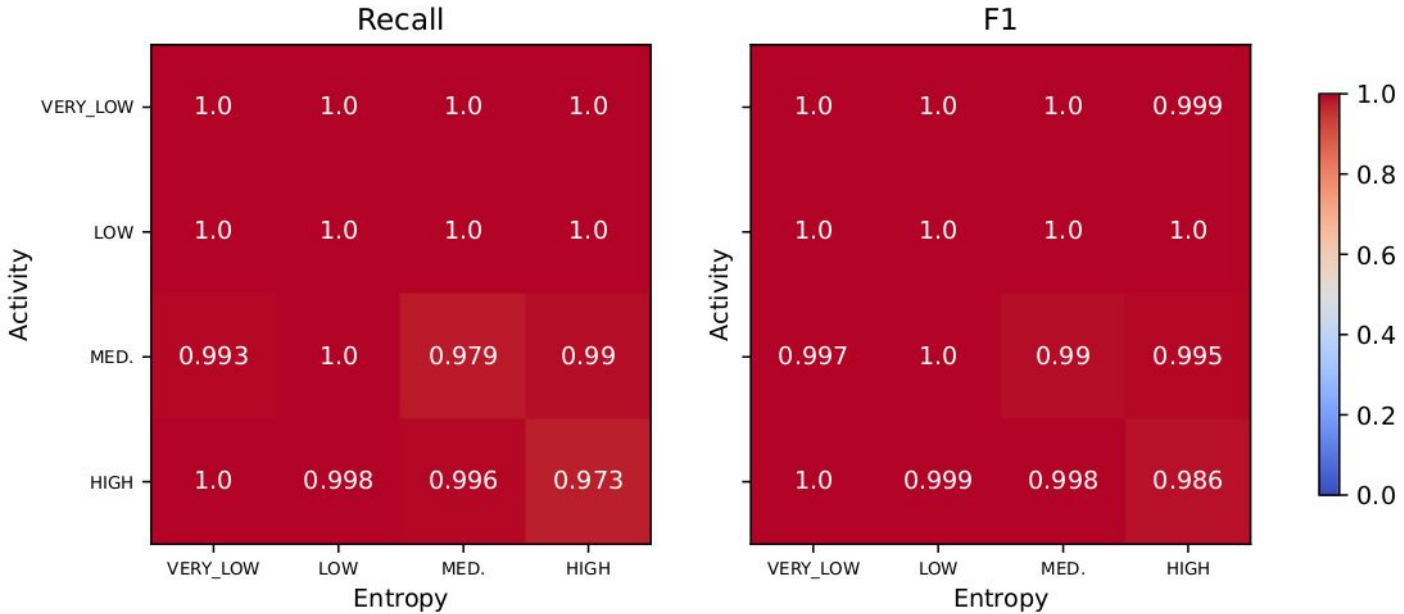
In the spirit of science, there really is no such thing as a 'failed experiment.' Any test that yields valid data is a valid test.

Adam Savage

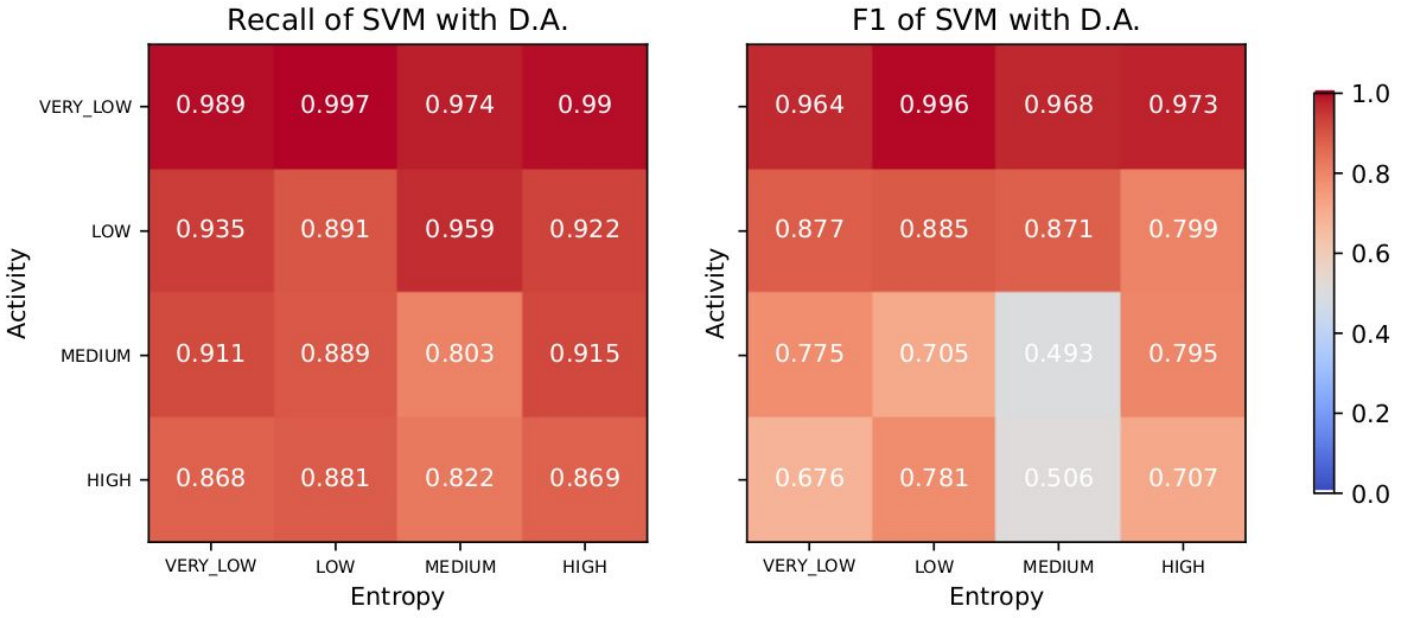
SVM: Architectural Counters Only



SVM: Full set of counters



Architectural Counters and Data Augmentation



Feature Selection

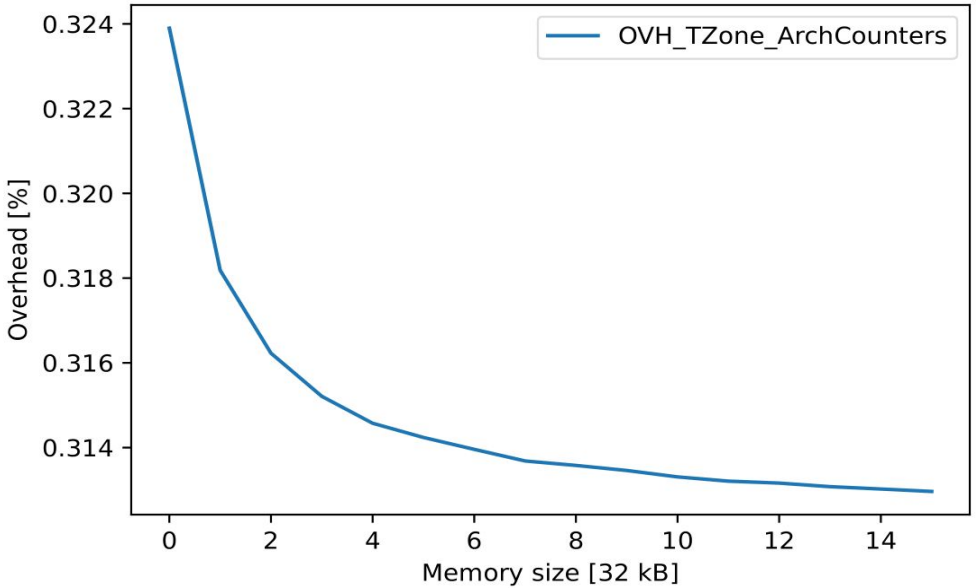
	DWT_LSUCNT	DWT_CPICNT	DWT_EXCCNT	DWT_CYCCNT	DWT_SLEEPCNT	DWT_FOLDCNT	LSU_stim	CPI_stim	FLD_stim	EXC_stim	TIME
(VERY_LOW,VERY_LOW)			X			X				X	
(VERY_LOW,LOW)			X			X				X	
(VERY_LOW,MEDIUM)			X			X				X	
(VERY_LOW,HIGH)			X			X				X	
(LOW,VERY_LOW)		X	X	X			X	X	X	X	
(LOW,LOW)	X			X			X	X	X	X	
(LOW,MEDIUM)	X	X	X	X		X	X				
(LOW,HIGH)	X		X	X				X	X		
(MEDIUM,VERY_LOW)	X	X	X				X				
(MEDIUM,LOW)	X	X	X	X		X	X		X		
(MEDIUM,MEDIUM)	X										
(MEDIUM,HIGH)	X		X			X	X				
(HIGH,VERY_LOW)	X	X	X	X			X			X	
(HIGH,LOW)	X	X	X								
(HIGH,MEDIUM)	X	X	X			X	X	X	X		X
(HIGH,HIGH)	X	X	X								

Overhead

Counters are **updated via hardware.**

Low overhead!

Overflow degradation: 32 bits prevent it



Overhead: Applicative Counters

Overhead depends on events.

Rough estimate if you know the average frequency of events

Some real examples:

1. Weather Monitor: frequencies from 0.3 to 2 Hz
2. Fall Detection Device: 32 Hz
3. Parkinson's Disease Monitor: 50 Hz

Conclusions

Life is the art of drawing sufficient conclusions from insufficient premises

Samuel Butler

G1

We claim that G1 was achieved.

Good performance, even without applicative counters.

Adding them improves classifiers

High Activity makes the problem much harder

G2

Architectural counters satisfy requirements: **OK**

Low-frequency applicative counters satisfy requirements: **OK**

Need high-frequency applicative counters: **KO**

Guidelines

Absolutely need low overhead? **Architectural counters**

Architectural counters provide low detection rate? **Enhance the Classifier**

Detection rate still too low? Happy with higher overhead? **Applicative counters**

Thanks for your attention!

Take time to be kind and to say ‘thank you’

Zig Ziglar