riscure

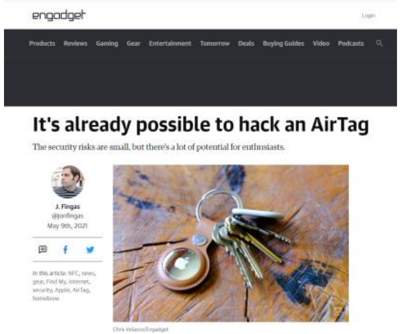
driving your security forward

HARDWARE SHIFT-LEFT

PRE-SILICON FAULT INJECTION EVALUATION AND POWER SIDE CHANNEL TESTING

NICOLE FERN

PRODUCTS IN THE WILD CONSTANTLY BEING HACKED USING FI AND SCA


engadget
Login
Products Reviews Gaming Gear Entertainment Tomorrow Deals Buying Guides Video Podcasts

It's already possible to hack an AirTag

The security risks are small, but there's a lot of potential for enthusiasts.

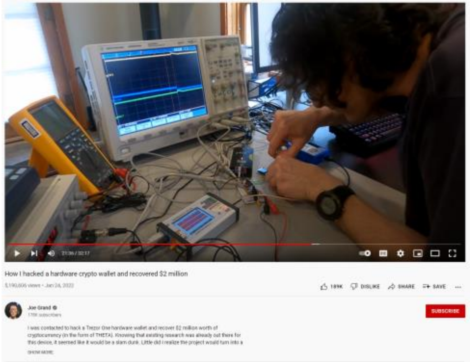
J. Fingas
@jonfingas
May 9th, 2021

In this article: NFC, news, gear, Find My, internet, security, Apple, AirTag, homebrew

Chris Velazco/Engadget

https://www.engadget.com/airtag-nfc-hack-205735409.html

http://www.ausgamestore.com/xbox-360-slim/matrix-glitcher-slim.html



How I hacked a hardware crypto wallet and recovered $2 million

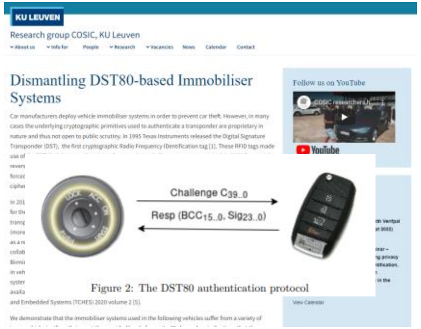
3,190,606 views • Jan 24, 2022

Joe Grand
170K subscribers

I was contacted to hack a Trezor One hardware wallet and recover $2 million worth of cryptocurrency (in the form of THETA). Knowing that existing research was already out there for this device, it seemed like it would be a slam dunk. Little did I realize the project would turn into a

SHOW MORE

https://www.youtube.com/watch?v=dT9y-KQbqi4



Research group COSIC, KU Leuven

Dismantling DST80-based Immobiliser Systems

Car manufacturers deploy vehicle immobiliser systems in order to prevent car theft. However, in many cases the underlying cryptographic primitives used to authenticate a transponder are proprietary in nature and thus not open to public scrutiny. In 1995 Texas Instruments released the Digital Signature Transponder (DST), the first cryptographic Radio Frequency IDentification tag [1]. These RFID tags made

and Embedded Systems (TCHES) 2020 volume 2 [5].

We demonstrate that the immobiliser systems used in the following vehicles suffer from a variety of

Figure 2: The DST80 authentication protocol

https://www.esat.kuleuven.be/cosic/blog/dismantling-dst80-based-immobiliser-systems/

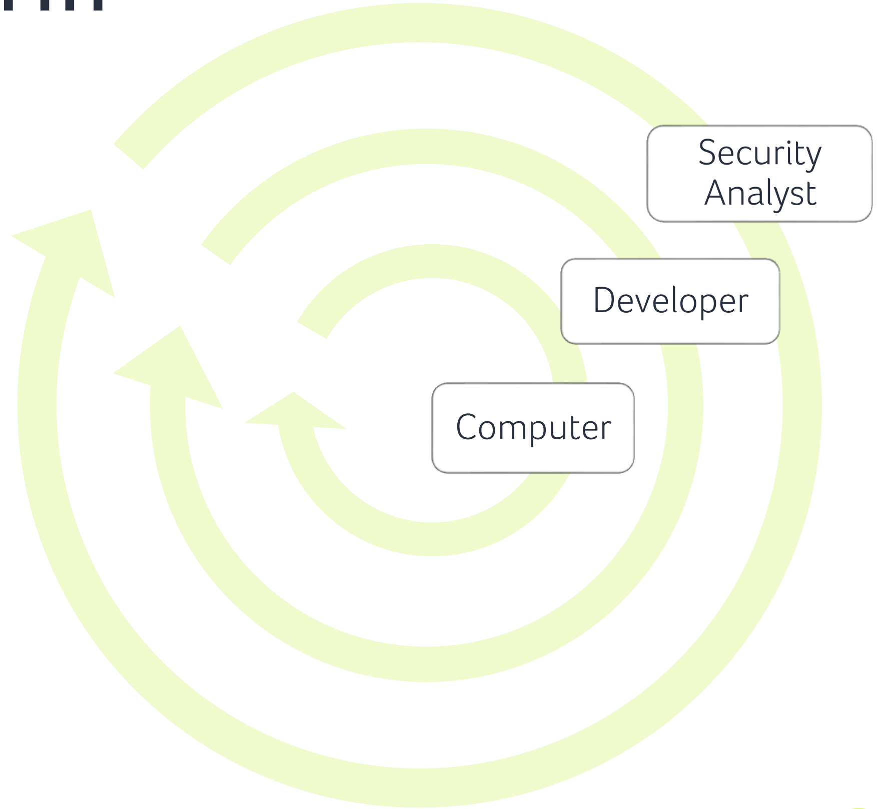
public

SECURITY TOOLING PHILOSOPHY

- Enable codifying of security insights that can integrate into continuous compute loop

- Enable developers/designers to perform security analysis

- An EDA tool with a light salting of security is not a security tool



GOAL OF THIS TALK

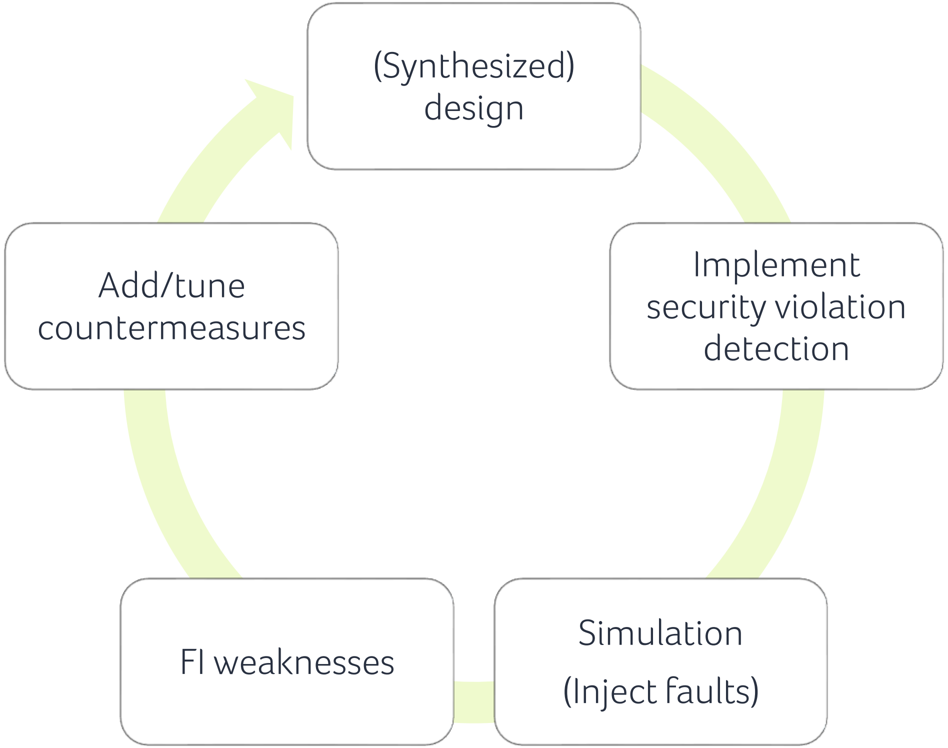
- Share our approach to infusing security knowledge into EDA tools and leveraging their best features for pre-silicon SCA/FI analysis
- Show some technical details and case studies behind our pre-silicon offerings
- Show pre-silicon SCA/FI analysis can be used as verification step to **detect** and **mitigate** issues before tapeout
- Show you don’t need to be a security expert to find issues
- Gather feedback

PRE-SILICON FAULT INJECTION

FI SIMULATION

1. Fault Model
   - Set of signals and time intervals faults injected during simulation

2. Outcomes
   - Classify effects of the fault as security relevant or not
   - Threat modelling required to implement **security violation detectors** (“assertions”)


(Synthesized) design

Implement security violation detection

Simulation (Inject faults)

FI weaknesses

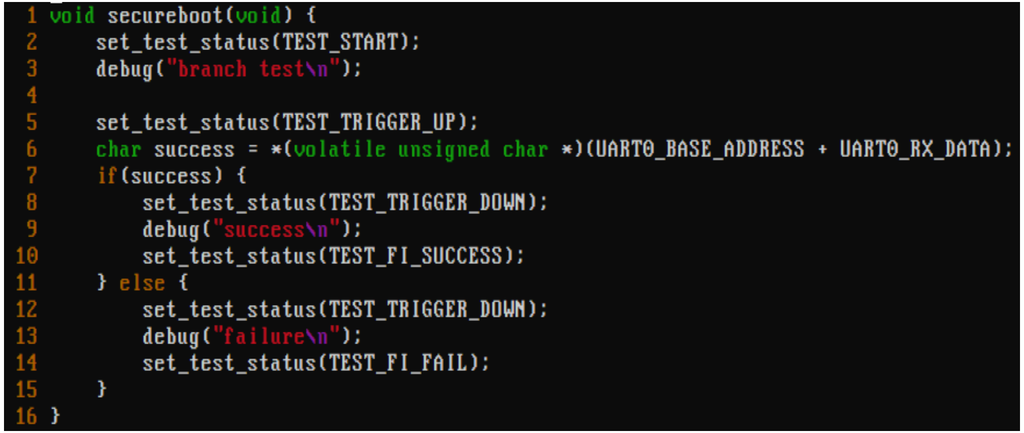
Add/tune countermeasures

CPU CASE STUDY: PICORV32

Identify sensitive RTL registers

- How can we flip a branch condition?

- Fault Model: single bit flips for all clocks and all registers/wires



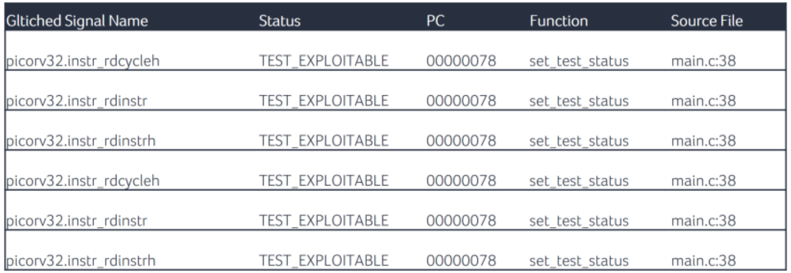
```
void secureboot(void) {
    set_test_status(TEST_START);
    debug("branch test\n");

    set_test_status(TEST_TRIGGER_UP);
    char success = *(volatile unsigned char *)(UART0_BASE_ADDRESS + UART0_RX_DATA);
    if(success) {
        set_test_status(TEST_TRIGGER_DOWN);
        debug("success\n");
        set_test_status(TEST_FI_SUCCESS);
    } else {
        set_test_status(TEST_TRIGGER_DOWN);
        debug("failure\n");
        set_test_status(TEST_FI_FAIL);
    }
}
```

https://github.com/YosysHQ/picorv32

RESULTS

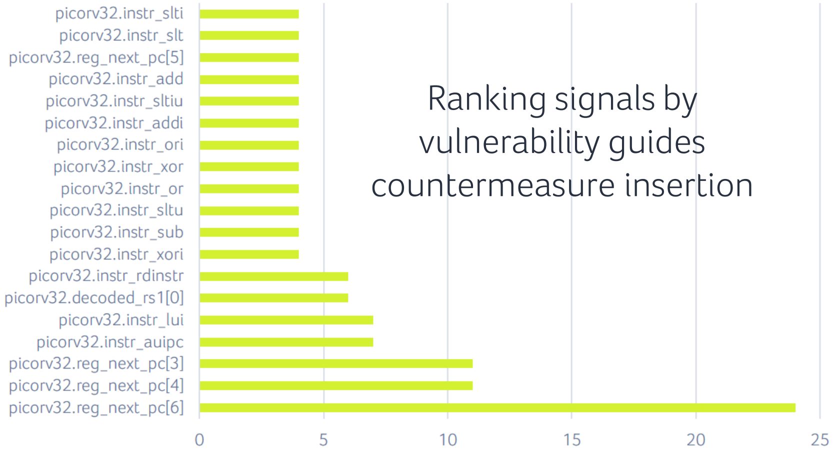
- Total faults injected: 1386400
- Exploitable: 408



Gltiched Signal Name	Status	PC	Function	Source File
picorv32.instr_rdcycleh	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38
picorv32.instr_rdinstr	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38
picorv32.instr_rdinstrh	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38
picorv32.instr_rdcycleh	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38
picorv32.instr_rdinstr	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38
picorv32.instr_rdinstrh	TEST_EXPLOITABLE	00000078	set_test_status	main.c:38

Vulnerable lines of code highlighted

Software countermeasures



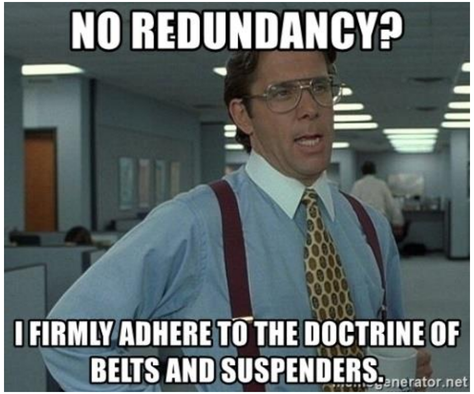
Ranking signals by vulnerability guides countermeasure insertion

# of exploitable faults

Hardware countermeasures

VERIFYING FI COUNTERMEASURES

- **HW countermeasure:** duplicate program counter and reset core if mismatch
- **SW countermeasure:** implement double check for success condition in software



	Vanilla	Harden RTL	Harden SW	Harden RTL + SW
Total	1386400	1540800	1386400	1540800
Exploitable faults	408	358	20	2
Exploitable signals	268	264	4	2
Detections	0	0	376	361

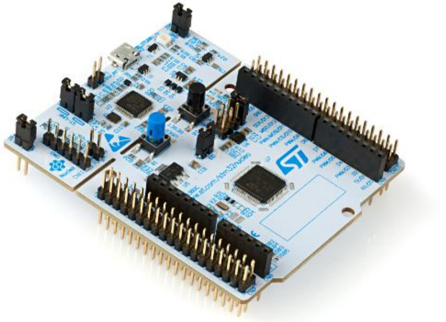
(this experiment is waaay to limited to conclude anything general about hardware vs software countermeasures)

ADDITIONAL CASE STUDIES

- OpenTitan AES round counter protection
  - FI simulation revealed that countermeasures inserted by OpenTitan team were optimized away by synthesis tool
  - Reported to OpenTitan team and they have a fix
  - https://github.com/lowRISC/opentitan

```
//////////////////////////////
// Round Counter Protection //
//////////////////////////////
```

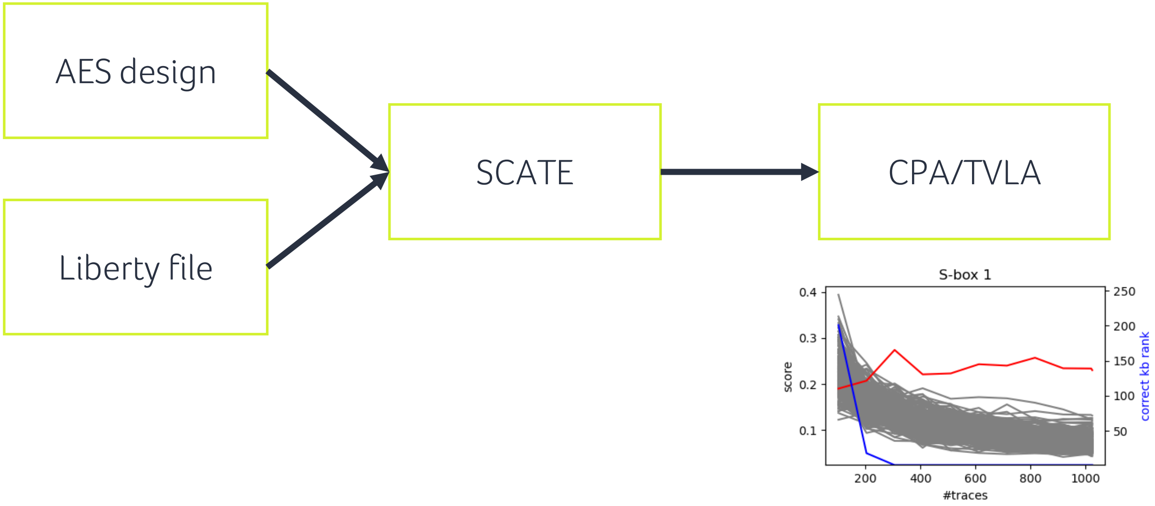
- Arm Cortex-M0
  - FI Simulator (netlist) v. EMFI on real device
  - There is a reasonable overlap with post-silicon testing



public

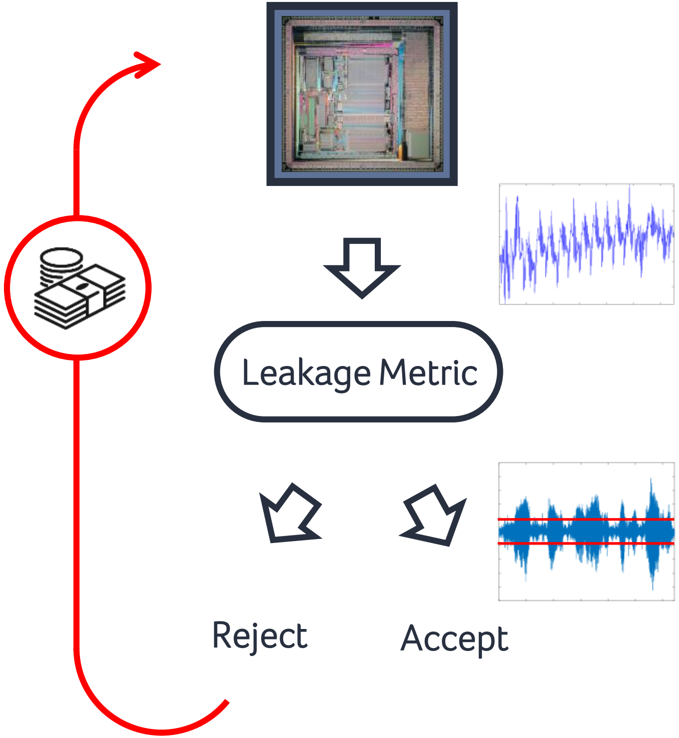
PRE-SILICON
SIDE CHANNEL ANALYSIS

SCA SIMULATION: SCATE



We can simulate traces to obtain “traditional” CPA plots – noiseless!

POST-SILICON

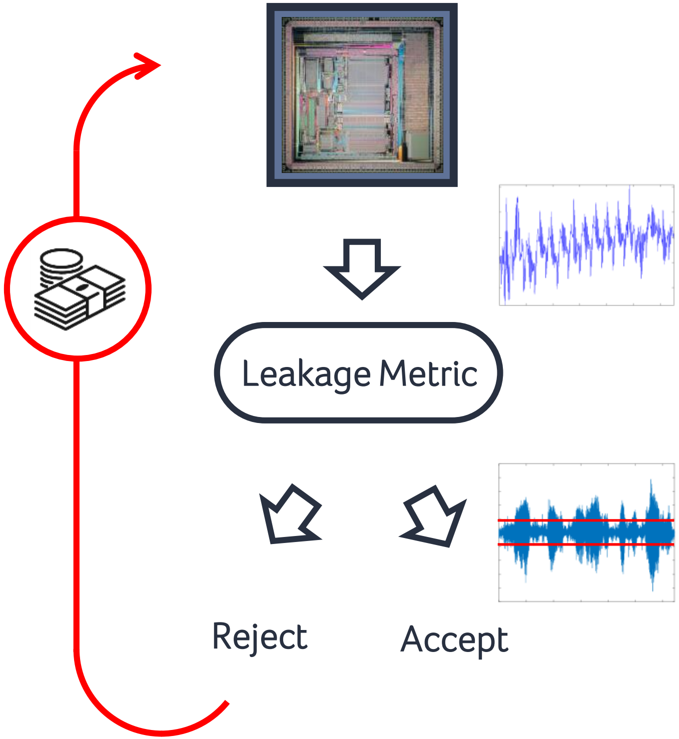


Leakage Metric

Reject

Accept

POST-SILICON

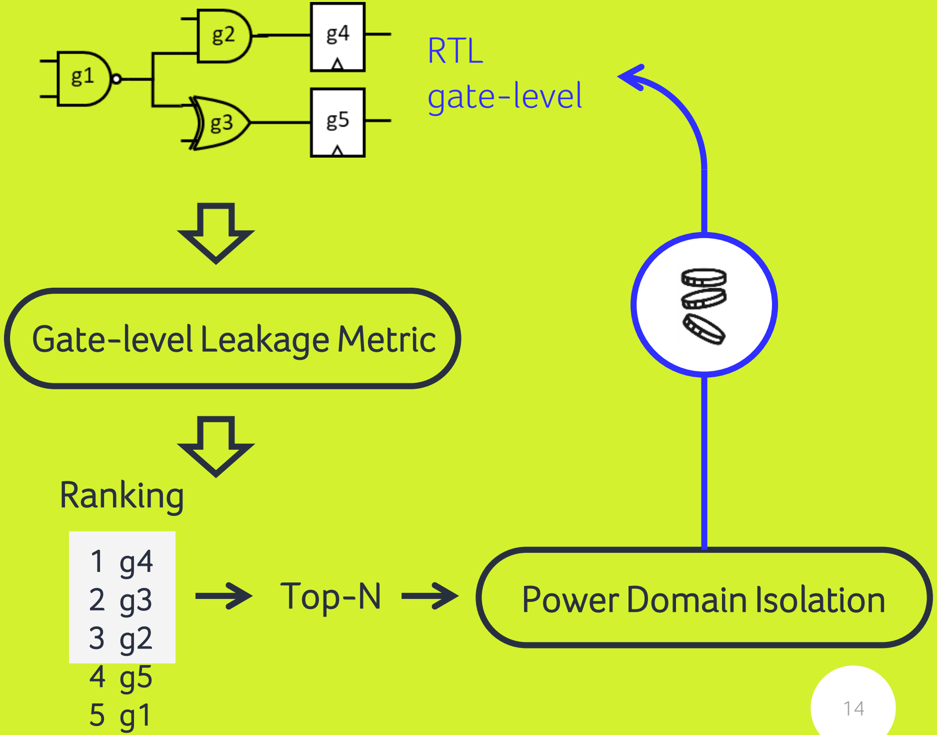


Leakage Metric

Reject

Accept

PRE-SILICON



RTL
gate-level

Gate-level Leakage Metric

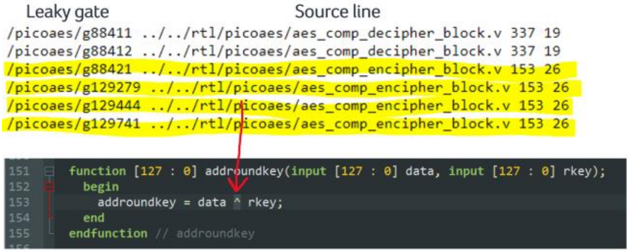
Ranking

1 g4
2 g3
3 g2
4 g5
5 g1

→ Top-N → Power Domain Isolation

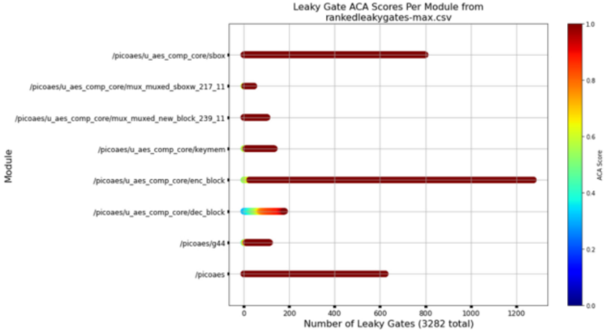
WITH RANKED LEAKY GATE LIST YOU CAN…

- Pinpoint root cause of leakage in design

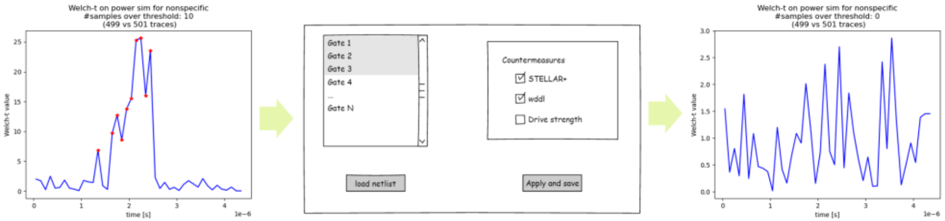


```
Leaky gate                       Source line
/picoaes/g88411 ../../rtl/picoaes/aes_comp_decipher_block.v 337 19
/picoaes/g88412 ../../rtl/picoaes/aes_comp_decipher_block.v 337 19
/picoaes/g88421 ../../rtl/picoaes/aes_comp_encipher_block.v 153 26
/picoaes/g129279 ../../rtl/picoaes/aes_comp_encipher_block.v 153 26
/picoaes/g129444 ../../rtl/picoaes/aes_comp_encipher_block.v 153 26
/picoaes/g129741 ../../rtl/picoaes/aes_comp_encipher_block.v 153 26
```

```
151   function [127 : 0] addroundkey(input [127 : 0] data, input [127 : 0] rkey);
152     begin
153       addroundkey = data ^ rkey;
154     end
155   endfunction // addroundkey
```



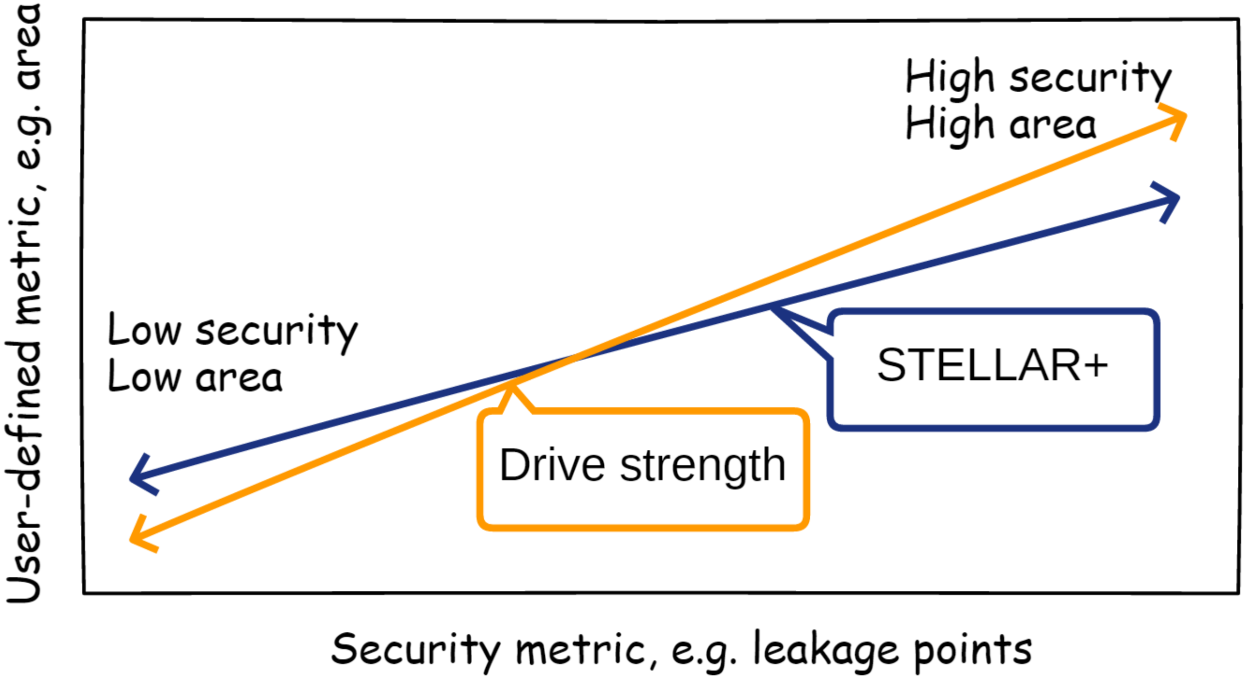
- Begin to automate countermeasure insertion (future work)



public

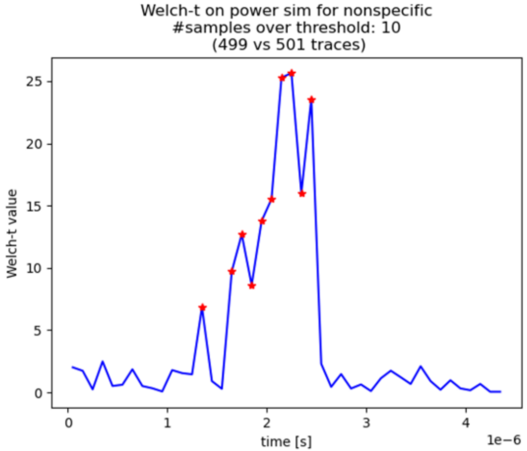
DESIGN SPACE EXPLORATION

For top N leaky gates gates, trade off security / user defined function / countermeasure


User-defined metric, e.g. area
High security
High area
Low security
Low area
STELLAR+
Drive strength
Security metric, e.g. leakage points

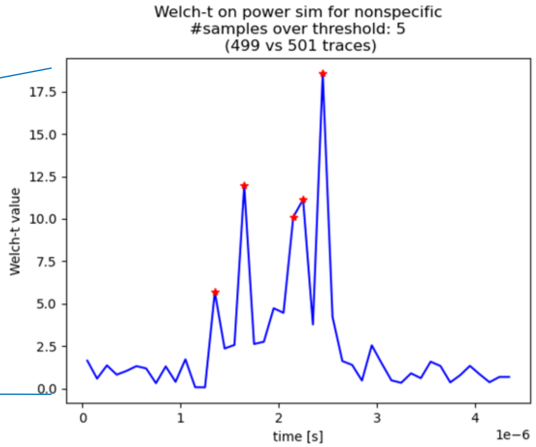
CASE STUDY: MASKED AES DESIGN

No masking



$x \longrightarrow y$

With masking

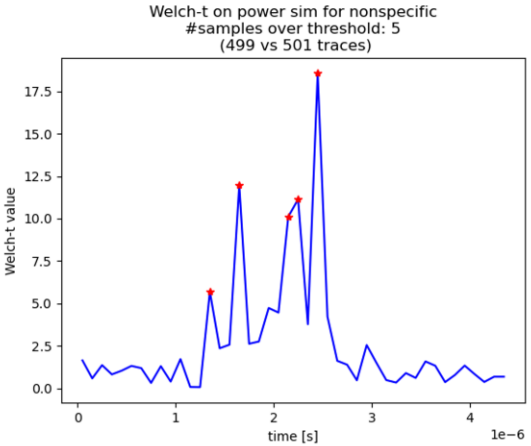


$x \oplus m_1 \longrightarrow y \oplus m_2$

$((x \oplus m_1) \oplus m_1) \oplus m_2 \longrightarrow y \oplus m_2$

CASE STUDY: MASKED AES DESIGN

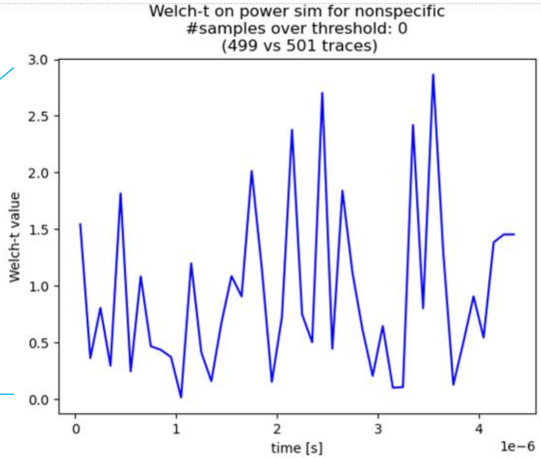
Masking – unconstrained synthesis



$x \oplus m_1 \rightarrow y \oplus m_2$

$((x \oplus m_1) \oplus m_1) \oplus m_2 \rightarrow y \oplus m_2$

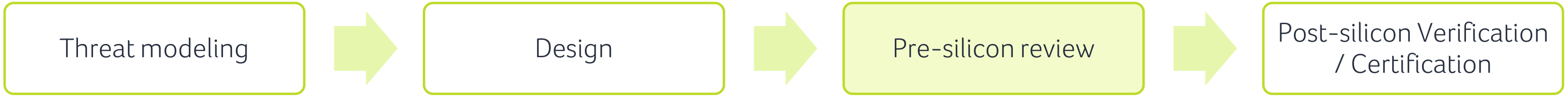
Masking – constrained synthesis



$x \oplus m_1 \rightarrow y \oplus m_2$

$((x \oplus m_1) \oplus m_2) \oplus m_1 \rightarrow y \oplus m_2$

KEY TAKEAWAYS



Threat modeling

Design

Pre-silicon review

Post-silicon Verification / Certification

> “With post-silicon we can fix the next chip, with pre-silicon we can fix the current chip.”
>
> – Customer Quote

We are:

Enabling **non-security-expert** designers **to root cause** security issues

Enabling continuous integration of insight and experience from security experts into the tooling

Reducing the need for **time consuming / expensive** post-silicon testing

Reducing the risk of post-silicon **certification failure** and **insecure** products

“Good security tools find potential vulnerabilities. Great security tools find practical vulnerabilities.”

– Jasper Van Woudenberg
(CTO, Riscure North America)

riscure

driving your security forward

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90
www.riscure.com

Riscure North America
550 Kearny St., Suite 330
San Francisco, CA 94108 USA
Phone: +1 650 646 99 79
inforequest@riscure.com

Riscure China
Room 2030-31, No. 989, Changle Road,
Shanghai 200031
China
Phone: +86 21 5117 5435
inforcn@riscure.com

riscure

driving your security forward