DESIGN, AUTOMATION & TEST IN EUROPE
14 – 18 March, 2016 · ICC · Dresden · Germany
The European Event for Electronic System Design & Test

Malicious hardware logic detection based on combinatorial testing

P. Kitsos, D. Simos, K. Stefanidis, A.G. Voyiatzis
TEI Western Greece (GR), ISI/R.C. “Athena” (GR), SBA Research (AT)

TRUDEVICE 2016 Workshop

Threat model

- Cryptography implementation in hardware
  - Example: AES-128
- A *small* Trojan is injected at one stage
  - Somehow …
  - Too big or too complex logic: easily detected
- An input pattern controls the Trojan activation
  - Example: using a few plaintext or key bits
  - Data communication scenario
- How can we detect it? Using minimal #patterns?
  - Today: black-box functional testing

Efficient test pattern generation

- Optimize test suite size
  - Exhaustive search is not an option
  - Size: time per test (multiple states)
  - Coverage: percentage of all cases covered
- ATPG and MERO
  - Good coverage but large size
  - Models do not account for extra (Trojan) logic
- Testability signals: can be used to hide Trojan
- Assumption of rareness is also challenged

Problem statement

- **Assume *n* input signals and that a combination of *k<<n* input signals activates the Trojan**

- **Construct test suites (sets of test vectors) of minimal size *T(n,k)* that cover <u>all</u> possible *k*-subspaces**

- **But:**
  - **Given *n* and *k*: unknown how to derive $T(n, k)$**
  - **Even its size is unknown: $2^k \leq |T(n, k)| \leq 2^n$**

Solution approaches

- **Naïve approach: $\binom{n}{k} \times 2^k$ possible vectors**
  - **All possible binary combinations**
  - **Over all possible position combinations**
- **Constant-Weight Vector:**
  - **Full coverage**
  - **Big sizes**
- **Randomly-generated vectors:**
  - **Adjustable size**
  - **No coverage guarantee (<100%)**

Combinatorial testing

- Proven method for testing critical software systems
  - NIST rule-of-thumb: testing for up to 6-way interactions is “just good enough”
- Model the input space (number and size of variables), their constraints, and dependencies
  - In our case: simple, just plain binary inputs
- Guaranteed (100%) coverage of input space
  - All possible patterns are exercised
- Optimization techniques: reduce test suite size
  - Number of required tests
  - Important if multiple tests, many SUTs, or time-consuming setup of experiment (e.g., systems)

Results: Optimized test suites

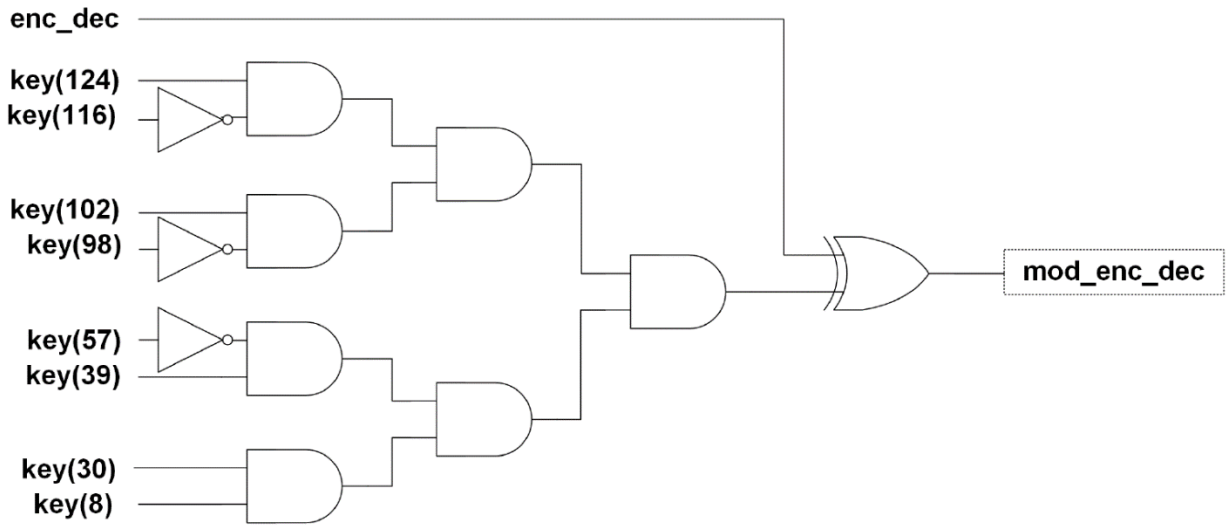
n	t	[22]	CWV [45]	ours	Covered patterns
128	2	2^7	129	11	32,512
128	3	-	256	37	2,731,008
128	4	2^{13}	8, 256	112	170,688,000
128	5	-	16, 256	252	8,466,124,800
128	6	-	349, 504	720	347,111,116,800
128	7	-	682, 752	2, 462	12,099,301,785,600
128	8	2^{23}	11, 009, 376	17, 544	366,003,879,014,400

$$\binom{128}{2} \times \mathbf{2}^2 = 32{,}512$$

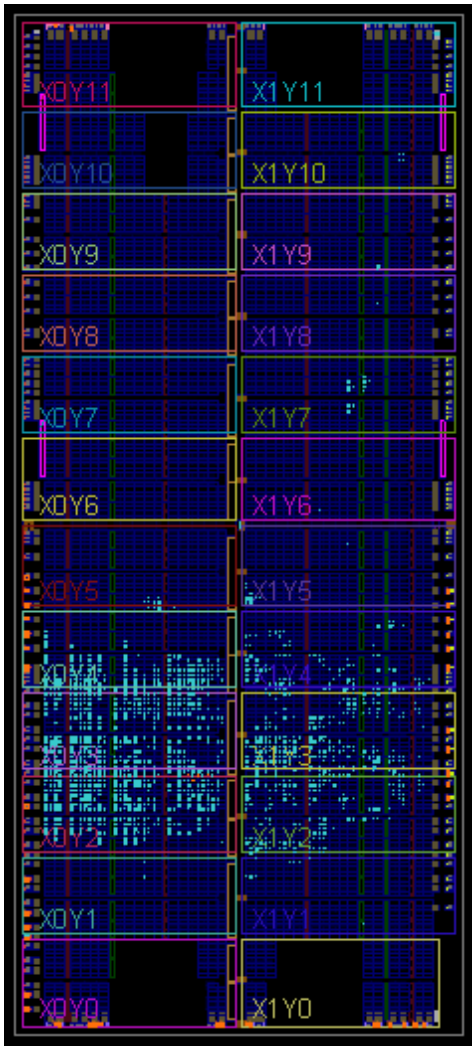
P. Kitsos, D.E. Simos, J. Torres-Jimenez, and A.G. Voyiatzis. *Exciting FPGA Cryptographic Trojans using Combinatorial Testing*. In 26th IEEE International Symposium on Software Reliability Engineering (ISSRE 2015). Gaithersburg, MD, USA, November 2-5, 2015.

Case study: A Trojan in AES-128

- Internals unknown (hint below ☺)
  - A simple combinational Trojan
  - Trojan monitors for specific key bit pattern
  - Trojan reverses the mode of operation (DoS)



AES + Trojan on Sakura-G board



Results: guaranteed activations

- **Test suite always activates Trojan**
  - **If test suite strength *t* > *k* monitored bits**
  - **Sometimes, even if *t* < *k* (partial success)**
- **At least one activation, exact # depends on pattern**

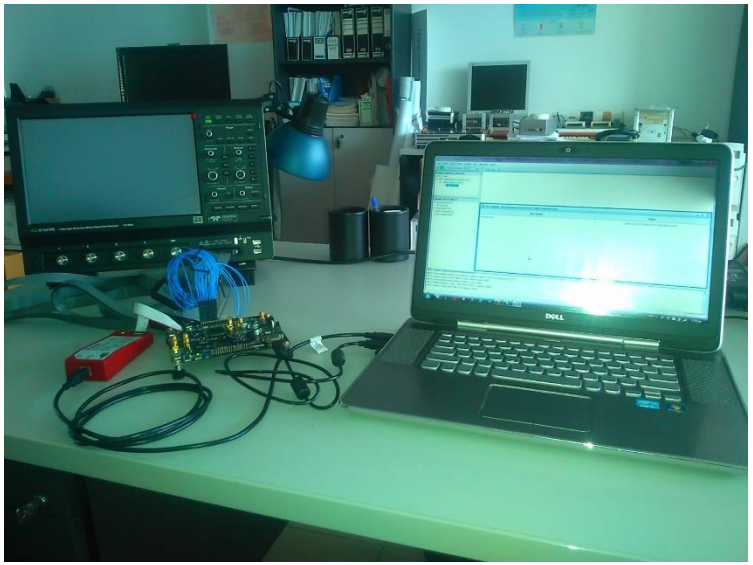
t	Suite size	Number of activations		
		$k = 2$	$k = 4$	$k = 8$
2	11	5	3	0
3	37	12	4	0
4	112	32	7	1
5	252	62	14	1
6	720	307	73	6
7	2462	615	153	10
8	17544	4246	1294	178

Take away

- Define an upper limit *k*; beyond that limit Trojan can be detected anyway
- Test for Trojan presence
  - Here, functional testing (black-box)
  - Combinatorial testing (CT) is a viable approach
- CT significantly reduces number of vectors
  - From millions to a few thousands
- CT guarantees at least one activation
  - Random testing can be close enough but not 100%; would you risk it?

Directions of work

- Improve test suite sizes targeting $k>8$
- Further reductions in test suite size for $k<9$
  - Some are optimal already
- Sequential Trojans: can we do better than multiple applications of test suite?
  - Further reduction!
- Apply in non-functional test
  - Side channel?
  - Improved response?



Thank you!

TRUDEVICE 2016