FASTrust: Feature Analysis for Third-Party IP Trust Verification

Song Yao[1], Xiaoming Chen[2], Jie Zhang[3], Qiaoyi Liu[1], Jia Wang[4], Qiang Xu[3], Yu Wang[1], Huazhong Yang[1]

[1]Dept of EE Tsinghua University
Beijing 100084, China
{yaos11, liuqy13}@mails.tsinghua.edu.cn
{yu-wang, yanghz}@tsinghua.edu.cn

[2]ECE Dept, CMU
Pittsburgh, PA 15213, USA
xchen3@andrew.cmu.edu

[3]Dept of CSE CUHK
Shatin, N.T., Hongkong
{jzhang, qxu}@cse.cuhk.edu.hk

[4]Dept of ECE Illinois Institute of Technology
Chicago, IL 60616, USA
jwang@ece.iit.edu

Outline

Background

Hardware security
Cyber-Combat's First Shot

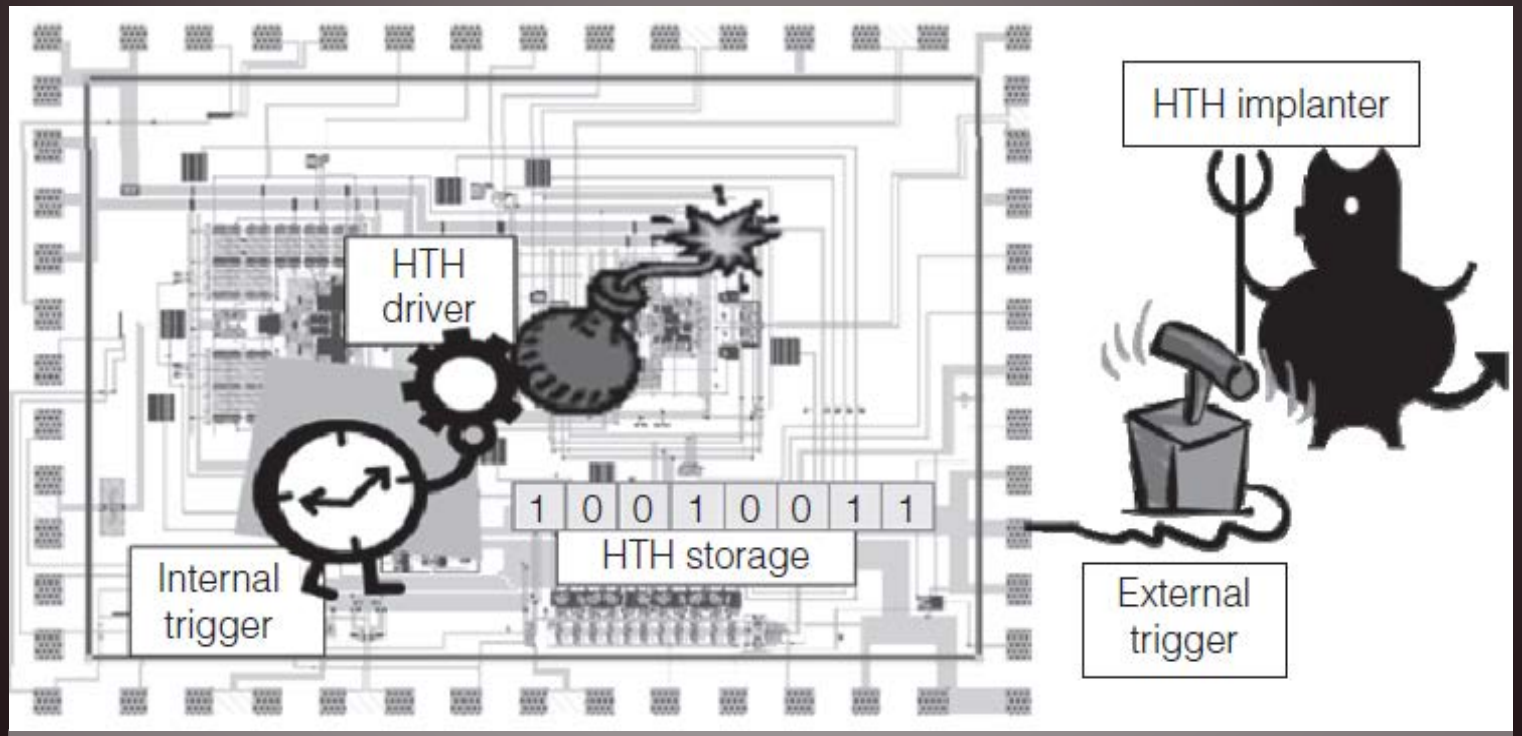
- In September 2007, Israeli jets bombed a suspected nuclear installation in northeastern Syria
- The mystery: failure of the Syrian radar system
- Speculation: The commercial microprocessors in the Syrian radar might have a hidden "backdoor"[1]
- An anonymous U.S. defense contractor: a "European chip maker" built into its microprocessors a kill switch that could be accessed remotely

[1] CYBER-COMBAT'S FIRST SHOT. By: Fulghum, David A., Wall, Robert, Butler, Amy, Aviation Week & Space Technology, 00052175, 11/26/2007, Vol. 167, Issue 21

Hardware Trojan

Kill Switch[2]—Hardware Trojan(HT)

- Focus—Pre-Silicon HT verification
- No golden model of IP or knowledge-base



[2] S. Adee, "The Hunt for the Kill Switch," IEEE Spectrum, vol. 45, no. 5, 2008, pp. 34-39.

Outline

Related Works

Redundancy-based Method

- Using several IPs from different vendors with the same functionality to implement runtime protection.

Knowledge-based Method

- Making used of the existing trusted circuit. Using that to identify the trusted and malicious circuit in an unknown IP by analyzing the RTL code or spec

Signal-based Method

- Considering no circuit structures, directly monitoring all signals, and determining whether a signal is suspicious based on the property of the signal itself

Circuit-based Method

- Considering circuit structures and detecting suspicious wires and circuits by checking dependencies between wires in combinational logic circuit blocks.

Existing Problem

FANCI [3]

Functional Analysis for Nearly Unused Circuit Identification

Truth table
Control Value

Veritrust [4]

Verification for Hardware Trust

Karnaugh Map
Malicious Onset Minterm
Malicious offset Maxterm

UCI [5]

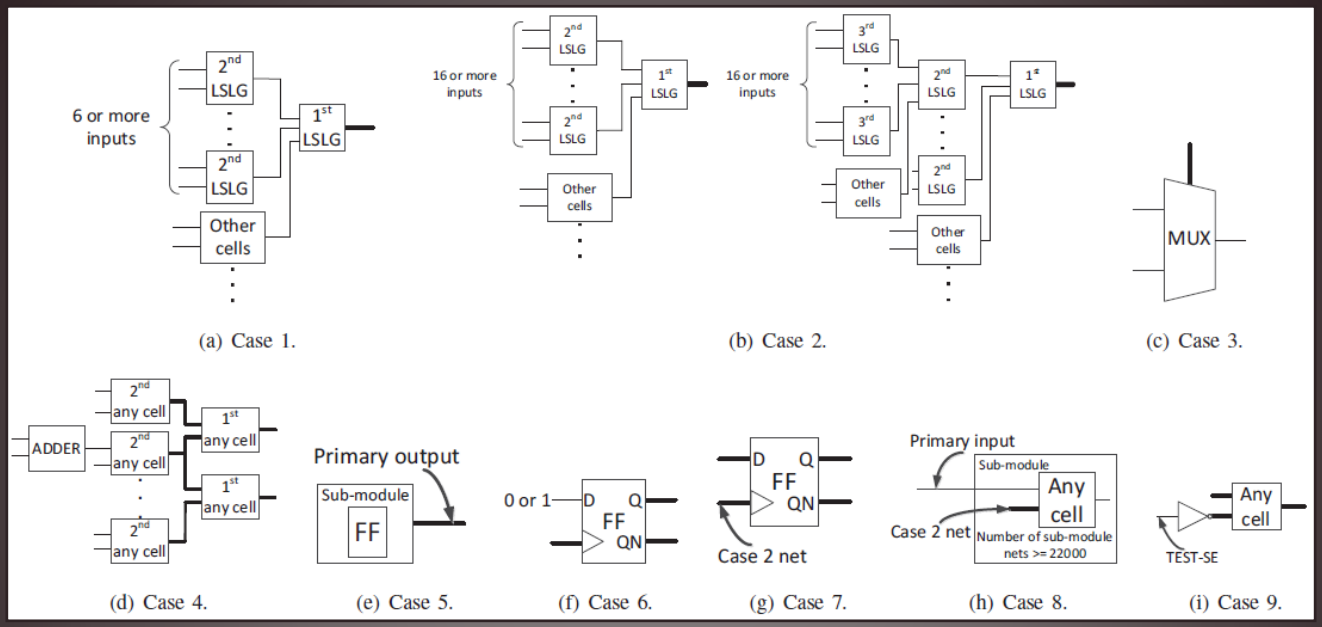
Unused Circuit Identification

Signal Pair
data flow graph

- High false positive v. Long runtime
- Low extensibility
- Analyzing combinational logic
- Incompetence on implicitly-triggered HTs

- [3] M. Hicks, M. Finnicum, S. King, M. Martin, and J. Smith, “Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in Security and Privacy (SP), 2010 IEEE Symposium on, May 2010, pp. 159 - 172.
- [4] J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu, “VeriTrust: Verification for hardware trust," in Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE, May 2013, pp. 1 - 8.
- [5] A. Waksman, M. Suozzo, and S. Sethumadhavan, “FANCI: identification of stealthy malicious logic using boolean functional analysis," in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM, 2013, pp. 697 - 708.

Feature Analysis related work



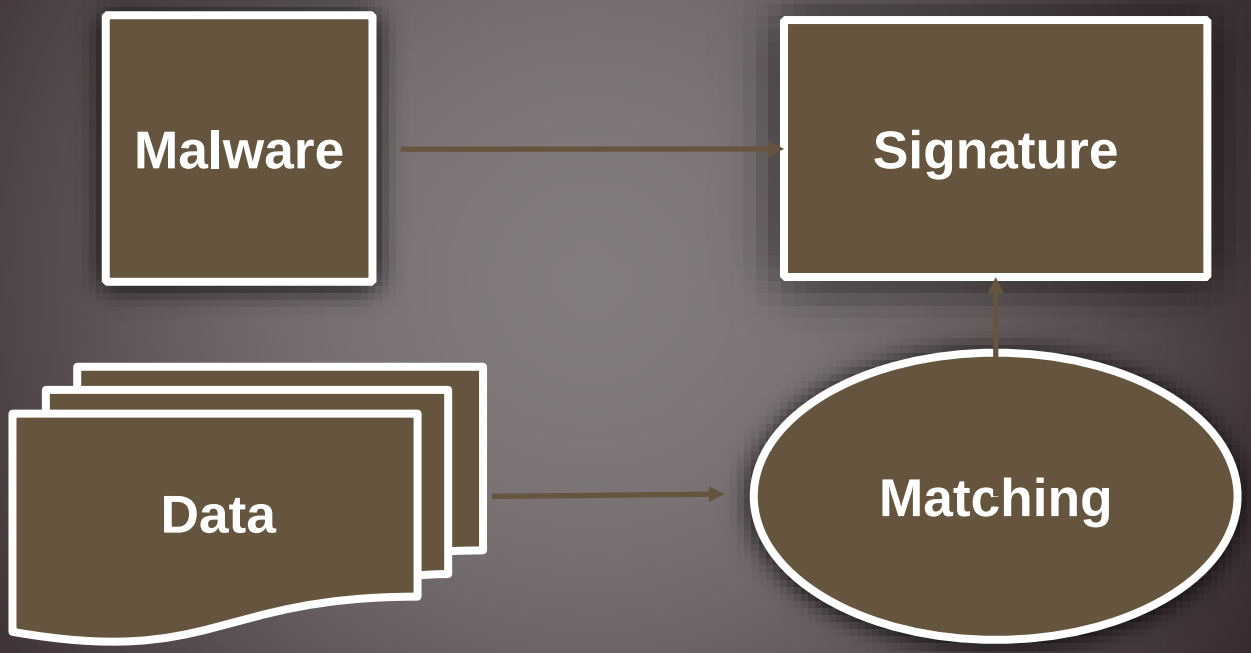
- Existing Problem
  - Based on Observations of Specific HT Design Cases, Not HT Taxonomy. No explanation why choosing this feature on circuit level.
  - Still Analyzing Circuit in Gate-Level
  - Time-consuming
  - Identify HT, cannot localize HT[6]

- [6] Oya M, Shi Y, Yanagisawa M, et al., “A score-based classification method for identifying hardware-trojans at gate-level netlists”, DATE 2015, pp. 465-470.

Outline

Motivation

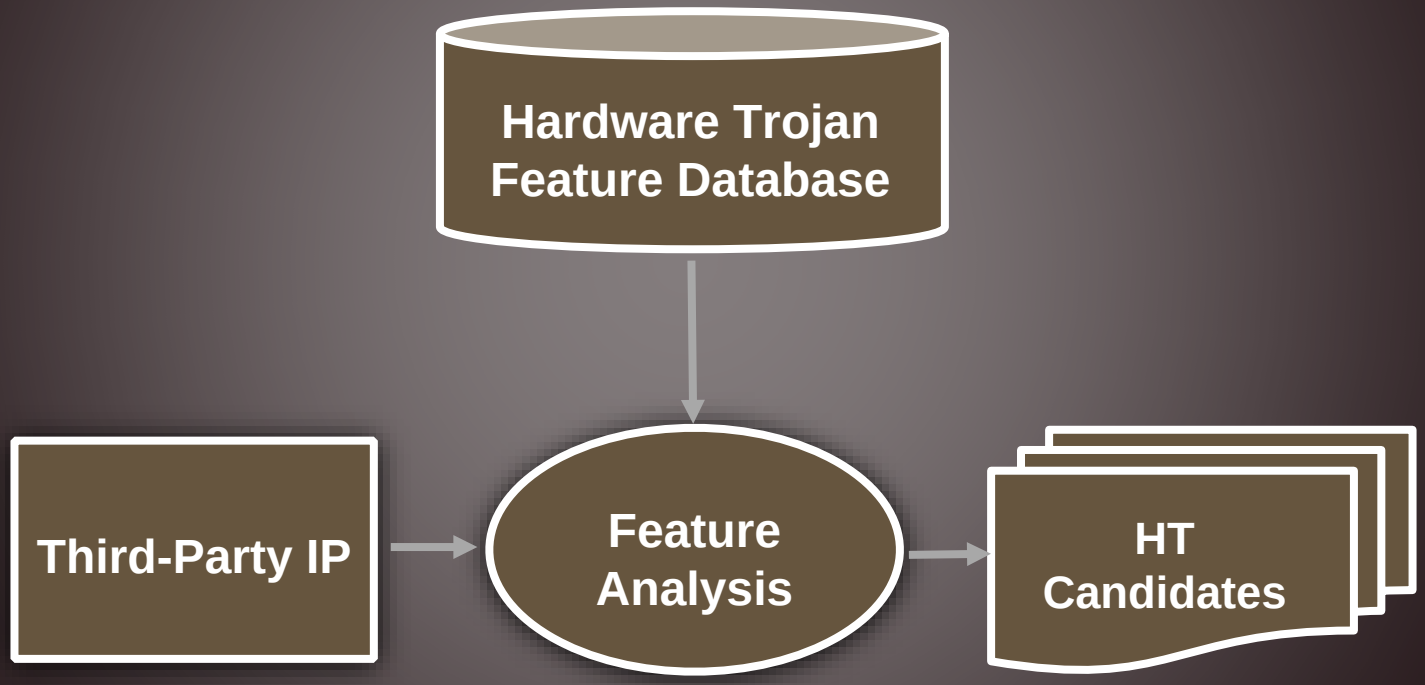
Malware Detection Framework



Detecting Hardware Trojans like Identifying Malware?

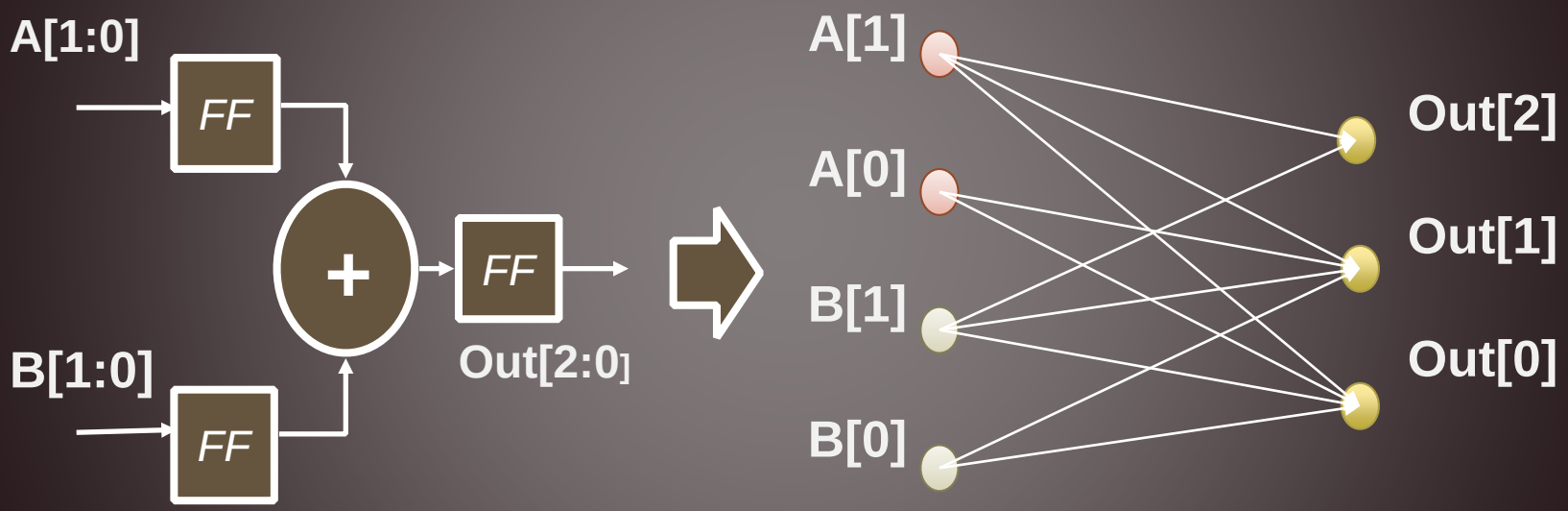
Feature Analysis

- Framework



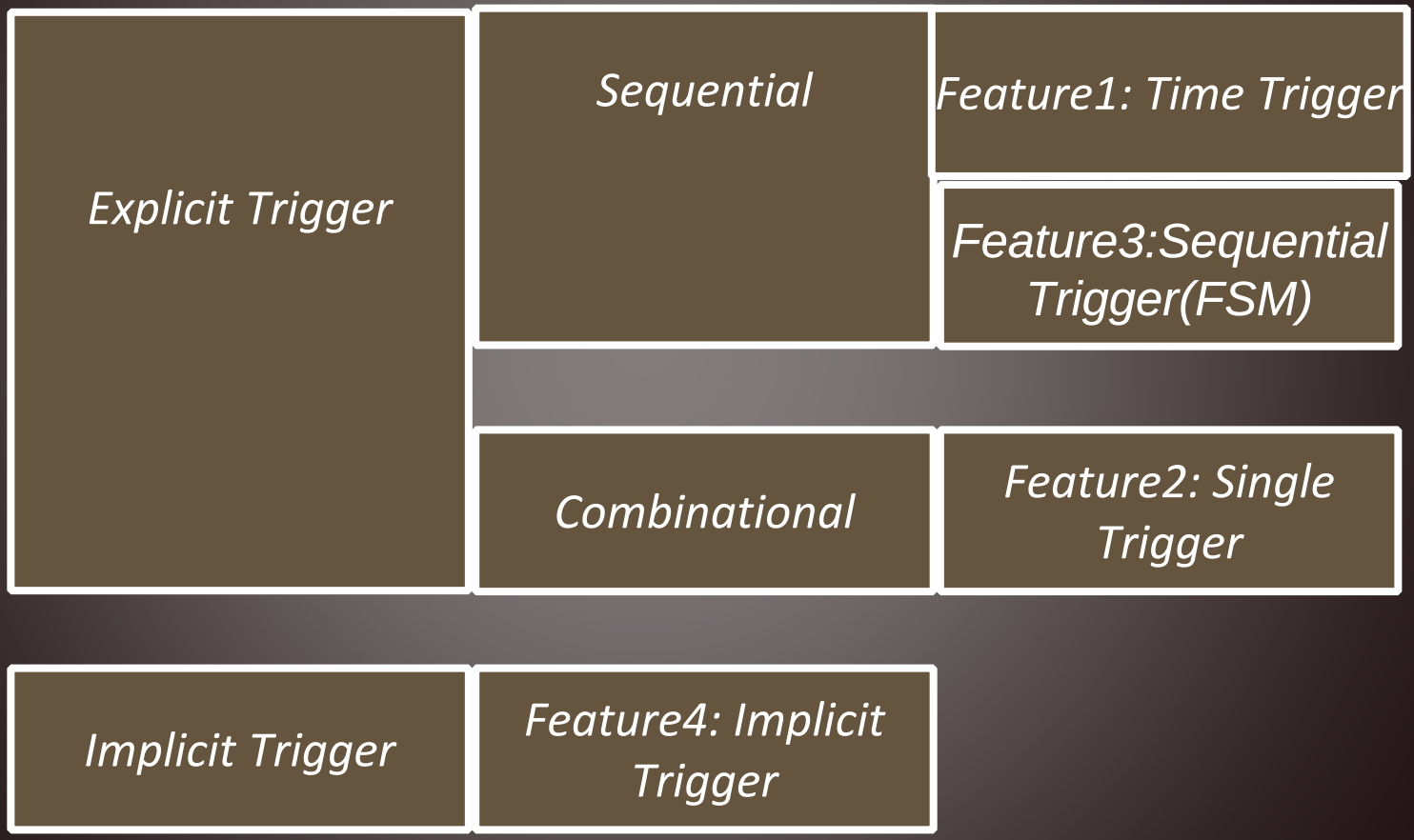
Tool to analyze features

Circuit Description in A New Level: Flip-Flop Level CDFG



- Use this simplify expression of the circuit to find the common nature on the CDFG (feature) among the different HT classified into the same category.

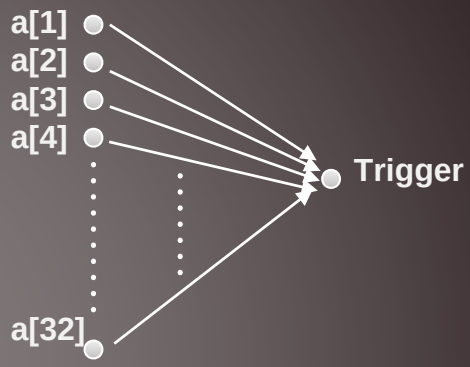
Taxonomy


Explicit Trigger
Sequential
Feature1: Time Trigger
Feature3:Sequential Trigger(FSM)
Combinational
Feature2: Single Trigger
Implicit Trigger
Feature4: Implicit Trigger

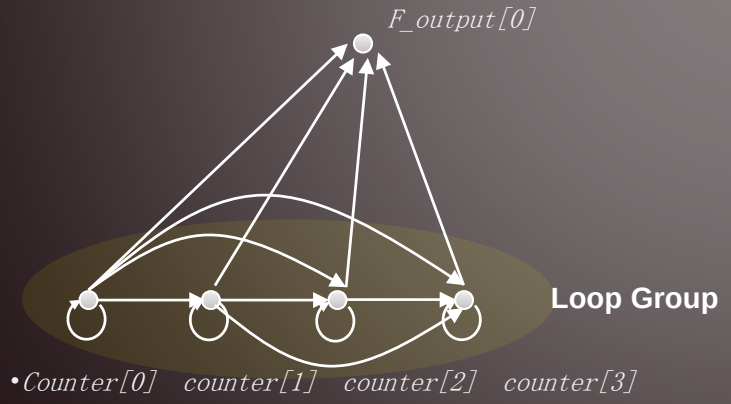
Feature Analysis Example 1

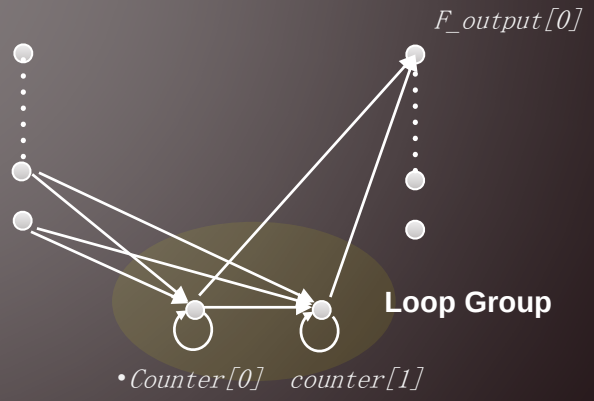
❖ Single-trigger Combinational HT — feature2

- Feature2: containing a node with extremely large in-degree.



❖ Sequential HT — feature1, 3



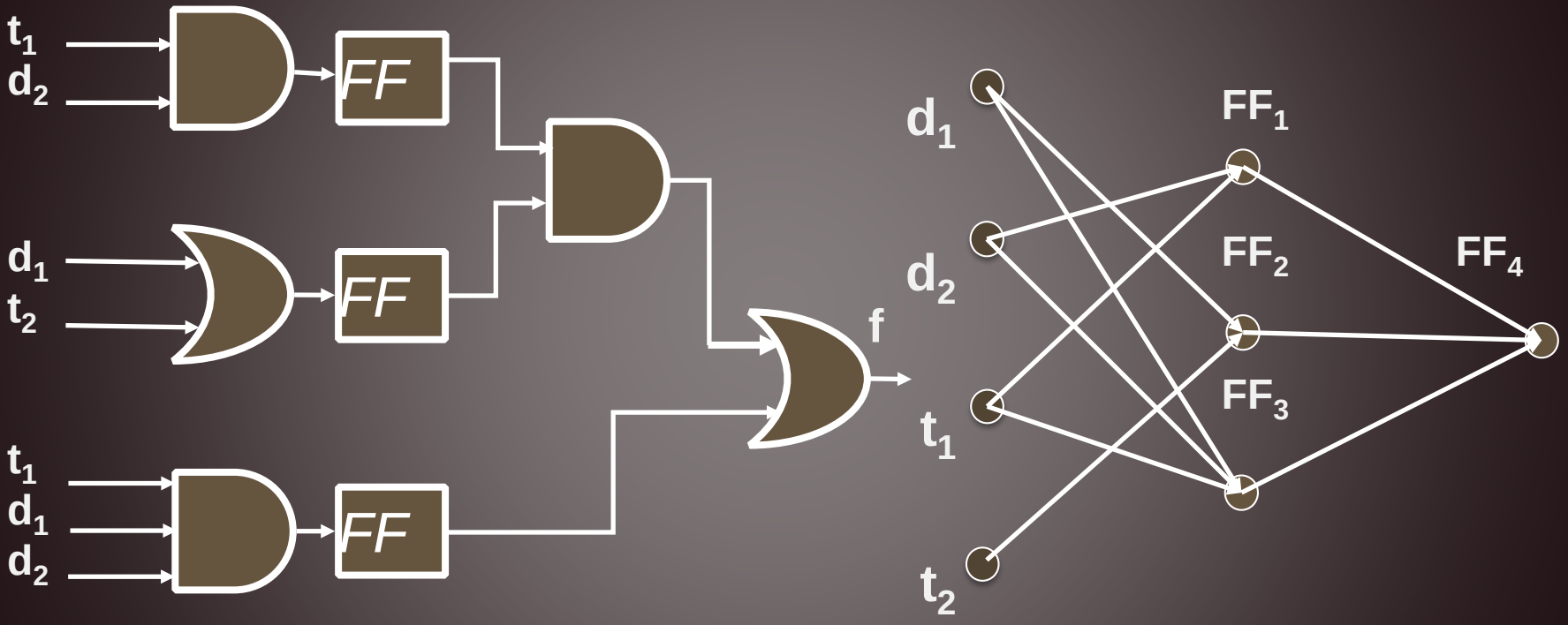


•Feature 1 : All nodes in the trigger circuit of a time-triggered HT form a large loop group.

•Feature 3: A sequential-triggered HT contains a loop group whose total in-degree from outside nodes is extremely large.

Feature Analysis Example 2

- Detecting An Implicitly-Triggered Hardware Trojan— Detrust



- Feature: Node with Out-Degree = 1

[7] J. Zhang, F. Yuan, and Q. Xu, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Pro- ceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. ACM, 2014, pp. 153–166.

Outline

Experiment Result

TABLE II
DETECTION CAPABILITY OF FASTRUST ON HARDWARE TROJANS

HT type	Benchmark	# of units (k)	# of nodes	Expected feature	Feature 1		Feature 2		Feature 3		Feature 4		Detected	Runtime		Peak memory (MB)
					Candidates	Hit	Candidates	Hit	Candidates	Hit	Candidates	Hit		Building CDFG (ms)	Feature analysis (ms)	
Time-trig.	RS232-T300	0.3	112	Feature 1	**1**	**1**	33	32	2	1	0	0	Yes	6.181	0.034	2.844
	RS232-T500	0.3	112		**1**	**1**	33	32	2	1	0	0	Yes	5.584	0.084	2.848
	BasicRSA-T300	2.2	718		**4**	**1**	297	15	4	1	0	0	Yes	190.1	0.098	18.15
	BasicRSA-T400	2.4	726		**5**	**1**	394	14	5	1	0	0	Yes	175.1	0.123	17.76
Single-trig.	BasicRSA-T100	2.3	693	Feature 2	4	0	**332**	**32**	4	0	0	0	Yes	165.2	0.113	17.02
	s35932-T100	6.0	2285		0	0	**1**	**1**	0	0	0	0	Yes	232.7	0.121	31.13
	s35932-T200	6.0	2284		0	0	**6**	**6**	0	0	0	0	Yes	238.3	0.120	31.06
	AES-T400	163.5	7391		1	0	**11**	**1**	1	0	1	0	Yes	3598	0.377	1011
	AES-T500	163.6	7391		1	0	**8**	**8**	1	0	1	0	Yes	3608	0.374	1011
	AES-T600	163.1	7236		0	0	**1**	**1**	0	0	0	0	Yes	3532	0.286	1010
	AES-T700	124.6	7255		0	0	**1**	**1**	0	0	1	0	Yes	2855	0.553	757.2
Seq.-trig.	PIC16F84-T100	1.3	534	Feature 3	0	0	125	0	**6**	**1**	17	0	Yes	11.82	11.77	8.004
	PIC16F84-T200	1.3	534		0	0	125	0	**6**	**1**	17	0	Yes	12.43	11.76	7.975
Impli.-trig.	s15850	10.6	773	Feature 4	3	0	168	0	8	0	**1**	**1**	Yes	12.35	0.885	59.80
	Wb_conmax	73.3	6646		1	0	1929	0	16	0	**2**	**2**	Yes	407.6	193.1	359.0
	Or1200_Ctrl	1.4	809		1	0	5	0	6	0	**2**	**2**	Yes	10.55	0.723	6.715

Feature 2 suffer from high false positive rate

Runtime from the experiment

[8] Trust-Hub benchmarks. https://www.trust- hub.org/resources/benchmarks.

Advantages

- Fast

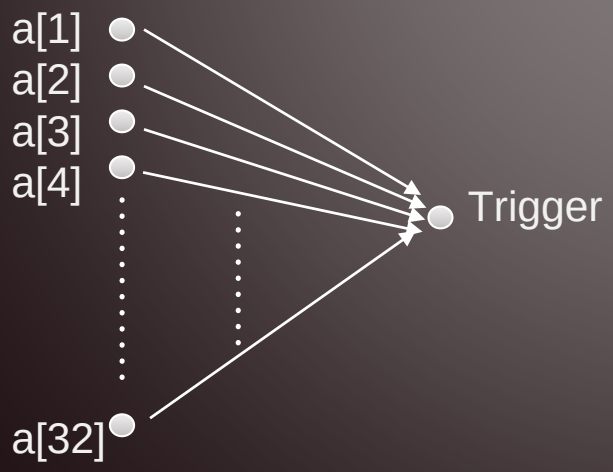
FANCI	Score-Based Feature Identification	FASTrust
From less than an hour till a couple of days	Approximately 3 hours	e.g.AES-T500 163.5k gate 7000nodes CDFG building: 3.608s Analysis: 0.377ms

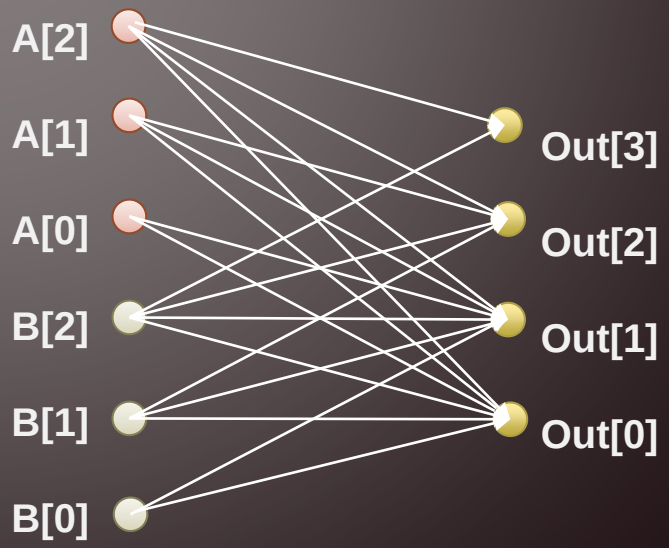
- No False Negative & Comparatively low False Positive
- Extensibility: New HT, New Feature
- Competence on Implicit-Triggered HTs

Limitation

- Threshold Concern
- False Positive Rate

- reasons:
  - Flip-Flop Level CDFG Lacks of Combinational Logic Information
  - HT shares same feature with the benign module





Outline

01 Background

02 Related Works

03 FASTrust

04 Experiment Result

05 Conclusion

Conclusion

- Novel 3PIP trust verification framework, named FASTrust
- High efficient and effective
- Extensibility
- First work to detect implicit-trigger HT

- Limitation
  - High false positive on feature 2
  - Threhold Concern

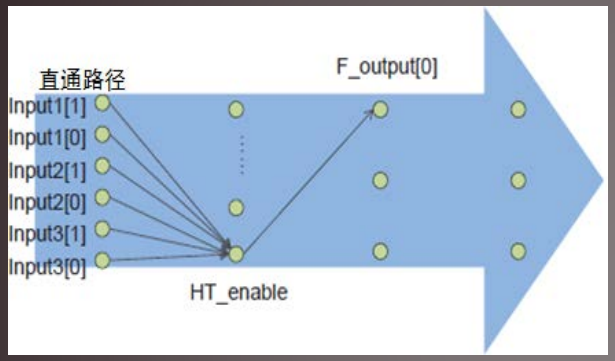
- Future Work

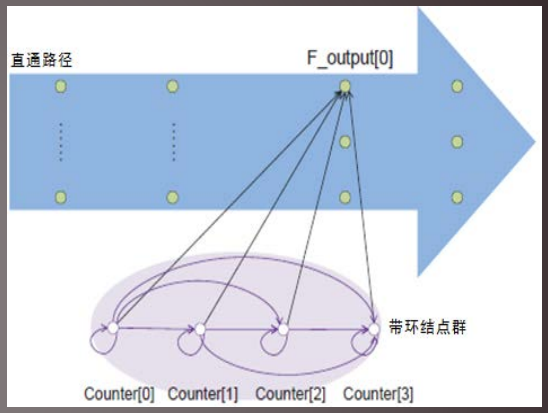
Thanks

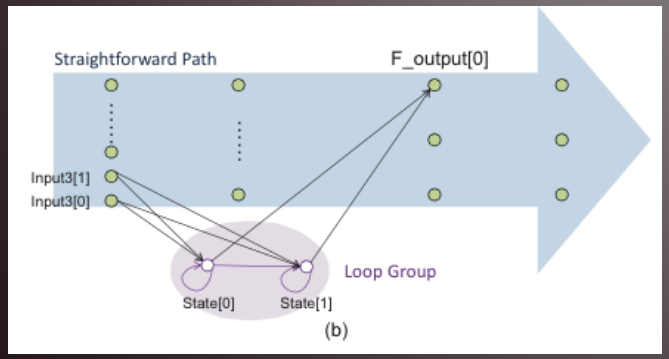
Q&A

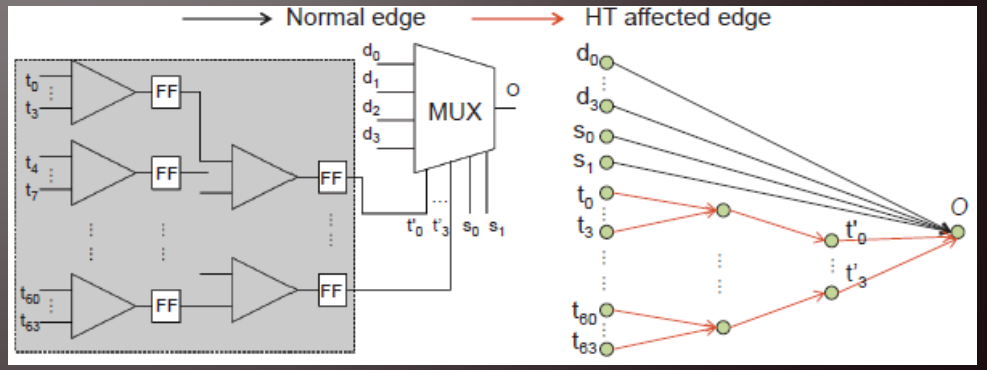
Back up

- Other features

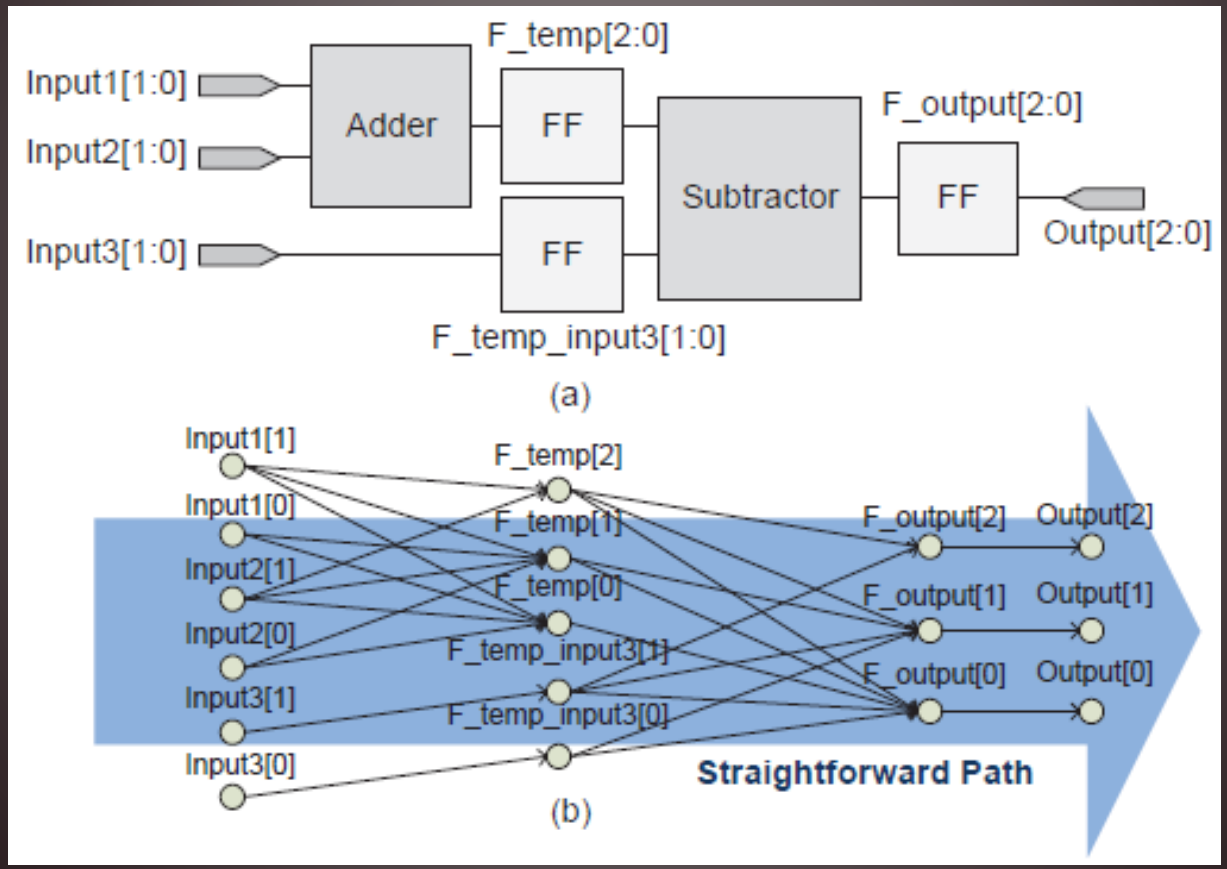








How to create register-level CDFG



Experiment Platform

FASTrust software is a single-thread program implemented in C++ language. The experiment platform is Ubuntu 14.04.1 LTS with an Intel Xeon E5-2690 CPU @2.90GHZ and 48GB RAM.

Benchmark	Threshold			
	1	2	3	4
s15850	1	14	26	30
Wb_conmax	2	17	16	16
Or1200_ctrl	2	12	15	18

TABLE I

THRESHOLDS FOR DIFFERENT FEATURES

	Feature 1	Feature 2	Feature 3	Feature 4
Threshold	20	20	50	1
Meaning	Node group size	Node's in-degree	Node group's total in-degree	successor number