RSAConference2019

San Francisco | March 4–8 | Moscone Center

BETTER.

SESSION ID: BAC-W02

Automated Fault Analysis of Block Cipher Implementations

Jakub Breier

Senior Cryptography Security Analyst
Underwriters Laboratories
Singapore
http://jbreier.com

Outline

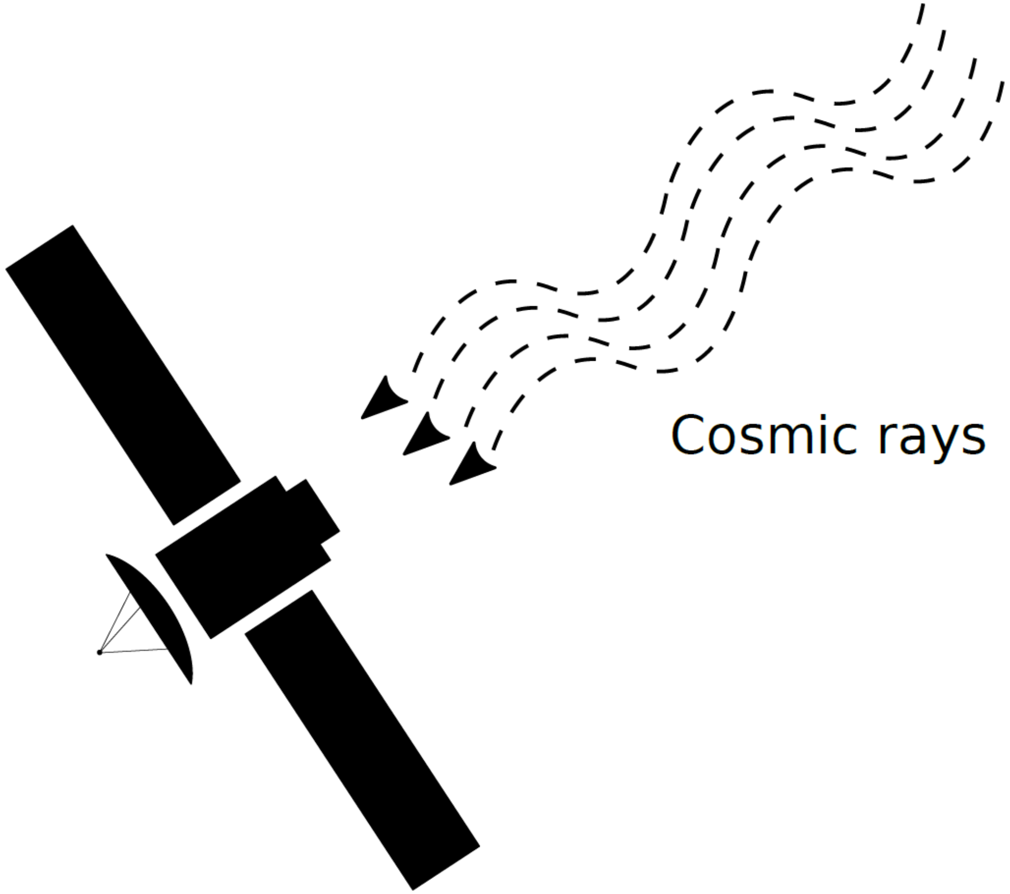
- Fault Analysis in Cryptography
- Differential Fault Analysis (DFA) of Symmetric Block Ciphers
- Automation of DFA for Software Implementations
- Countermeasure Implementation

Fault Analysis in Cryptography

Physical Attacks in Cryptography

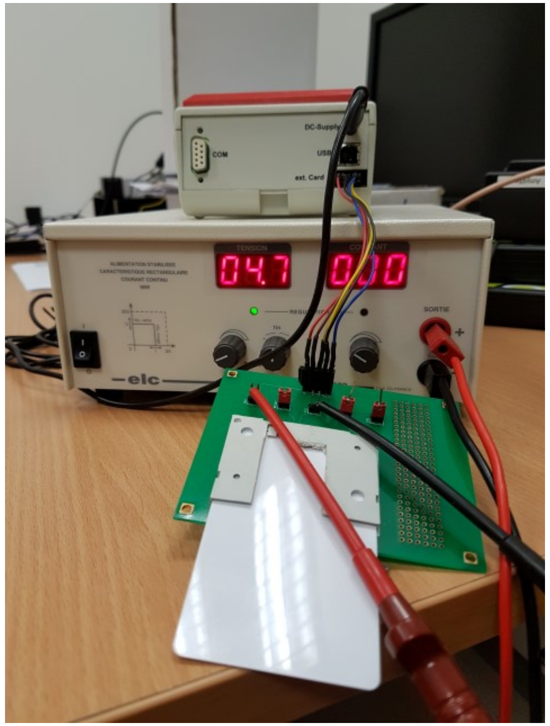
- Cryptography provides algorithms that enable secure communication in theory
- In real world, these algorithms have to be implemented on real devices:
  - software implementations - general-purpose devices
  - hardware implementations - dedicated secure hardware devices
- To evaluate security level of cryptographic implementations, it is necessary to include physical security assessment

First IC Disturbances – Cosmic Rays and Satellites



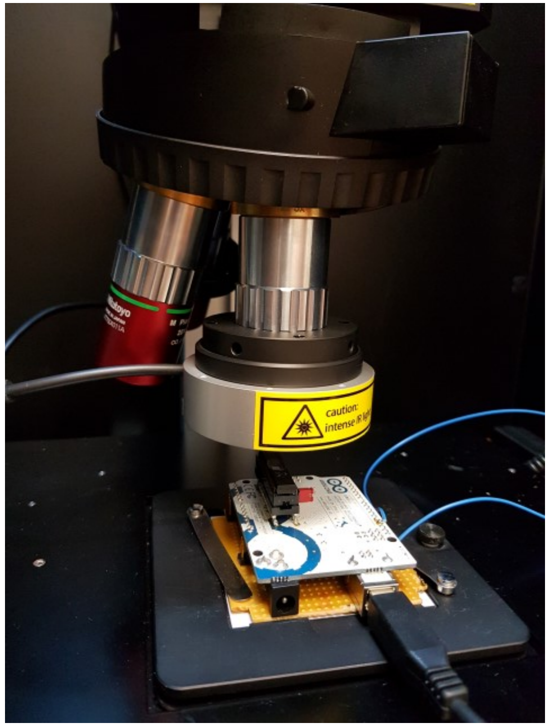
D. Binder et al. Satellite anomalies from galactic cosmic rays, IEEE Transactions on Nuclear Science, 1975.

Fault Injection Techniques in Practice



Voltage Glitching
$

EM Pulse Injection
$$



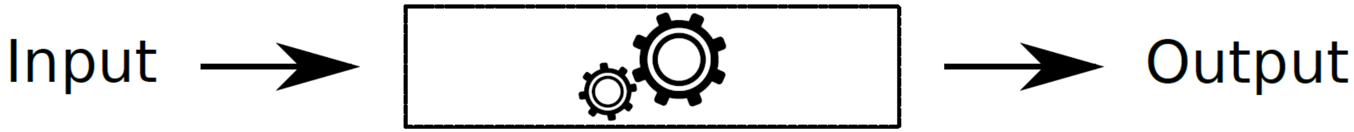
Laser Fault Injection
$$$

Why Fault Attacks?

- The best cryptanalysis of AES needs complexity of $2^{126.1}$

  - A. Bogdanov et al. Biclique cryptanalysis of the full AES, ASIACRYPT 2011.

- The best fault attack on AES needs just one faulty and one correct ciphertext pair



  - J. Breier et al. Laser Profiling for the Back-Side Fault Attacks: With a Practical Laser Skip Instruction Attack on AES, CPSS 2015.

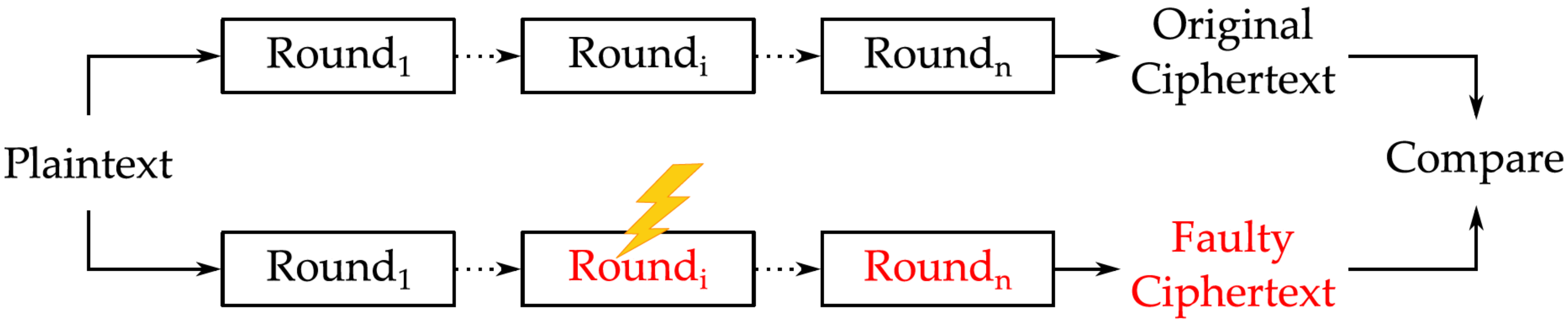
RSAConference2019

Differential Fault Analysis

of Symmetric Block Ciphers

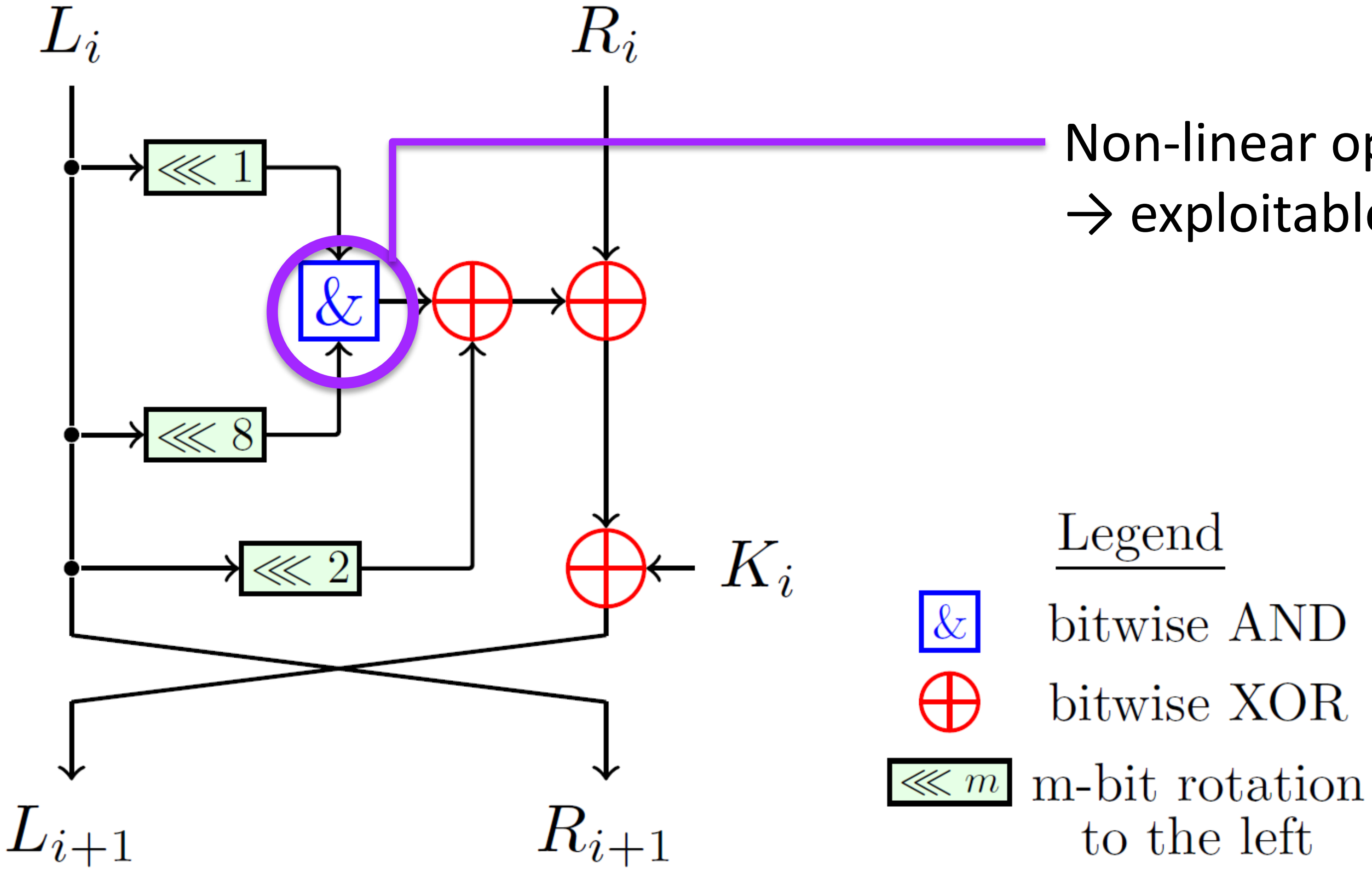
Working Principle

- Attacker injects a fault in a chosen round of the algorithm to get the desired fault propagation at the end of an encryption
- The secret key can then be determined by examining the differences between a correct and a faulty ciphertext



E. Biham and A. Shamir: Differential fault analysis of secret key cryptosystems, CRYPTO'97.

Example – SIMON Block Cipher



Non-linear operation
→ exploitable by DFA

R. Beaulieu et. al. The SIMON and SPECK Families of Lightweight Block Ciphers, ePrint 2013/404.

Exploiting AND Operation by DFA

a	b	c = a & b
0	0	0
0	1	0
1	0	0
1	1	1

Flip bit ‘a’ →

a’	b	c’= a’& b
1	0	0
1	1	1
0	0	0
0	1	0

- If the result does not change → ‘b’ is 0
- If the result changes → ‘b’ is 1

DFA - Discussion

- Different cipher families can be exploited by similar attack procedure, e.g.:
  - In SPN designs, Sbox is targeted
  - In ARX designs, modular addition is targeted
  - If a cipher uses MDS matrix, such as *MixColumns* in AES, this can be exploited for more efficient attack with lesser faults
- There is normally a trade-off between the computational complexity and the number of faults:
  - Last round attack – many faults, low complexity
  - $2^{nd}/3^{rd}$ last round attack – fewer faults, higher complexity

RSAConference2019

Automation of DFA for Software Implementations

Why Automation of DFA?

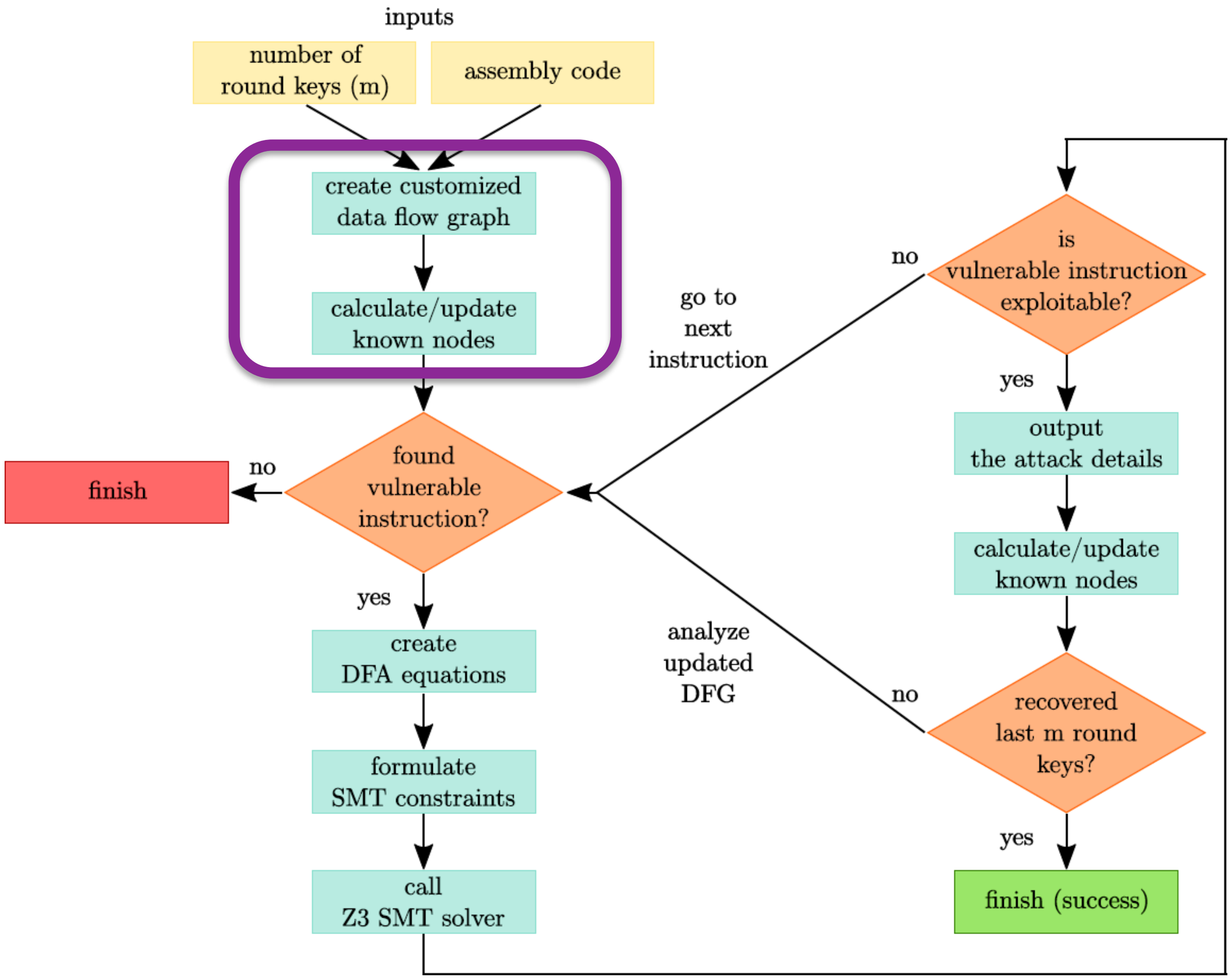
- All the current symmetric block ciphers have been shown vulnerable against fault attacks (especially DFA)
- The question is not whether the algorithm is secure or not, but which part of it is insecure
- Automated methods can provide an answer fast and with minimal need of human intervention

Tool for Automated DFA on Assembly – TADA

- The main idea – feed the assembly code to the tool and get the vulnerabilities, together with a way how to exploit them
- Static analysis module analyzes the propagation of the fault and determines what information can be extracted from known data
- SMT solver module solves the DFA equations, verifying whether an attack exists

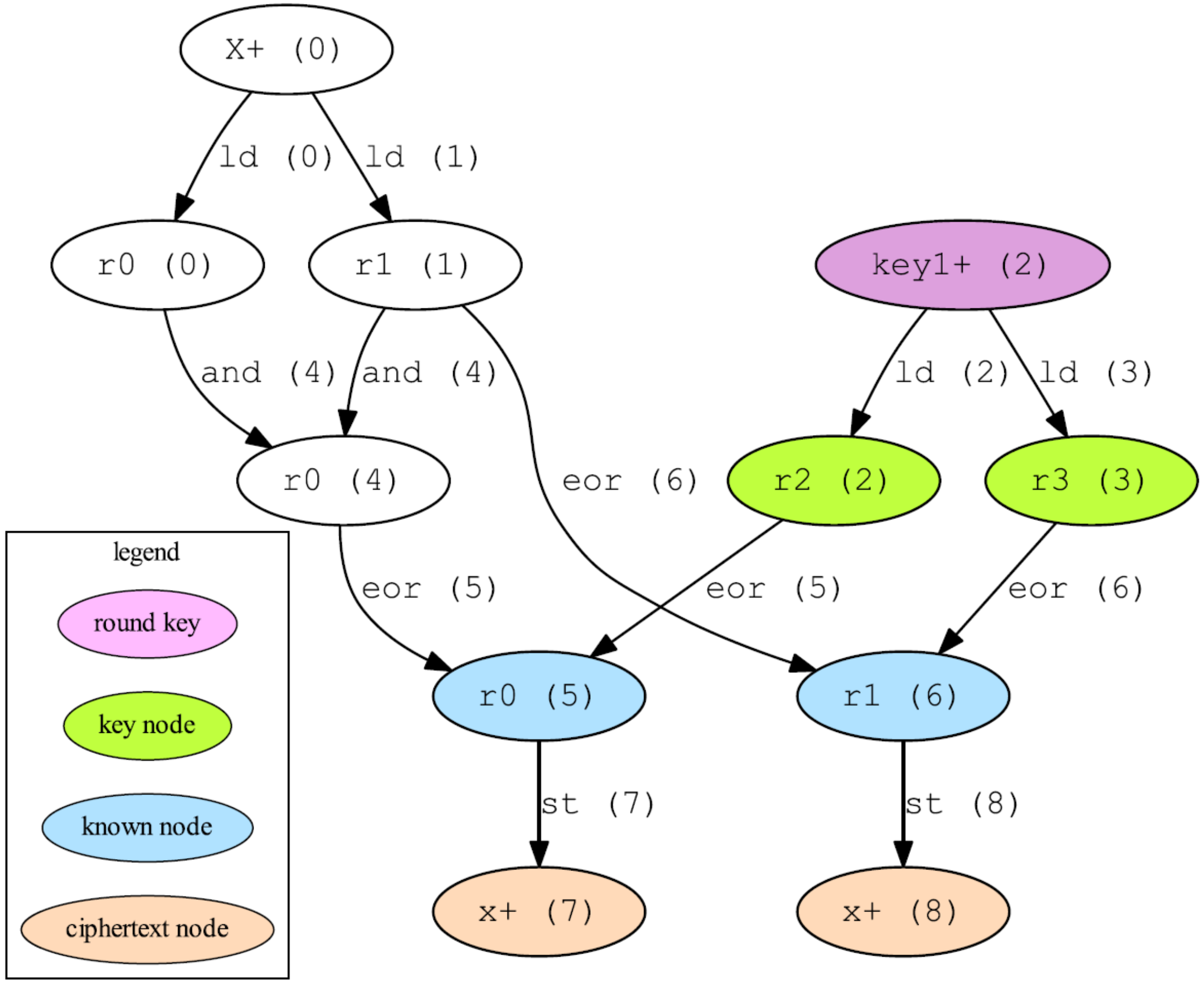
Analyze assembly file → Generate custom DFG → Construct DFA attack → Find the key

TADA – Detailed Process Flow

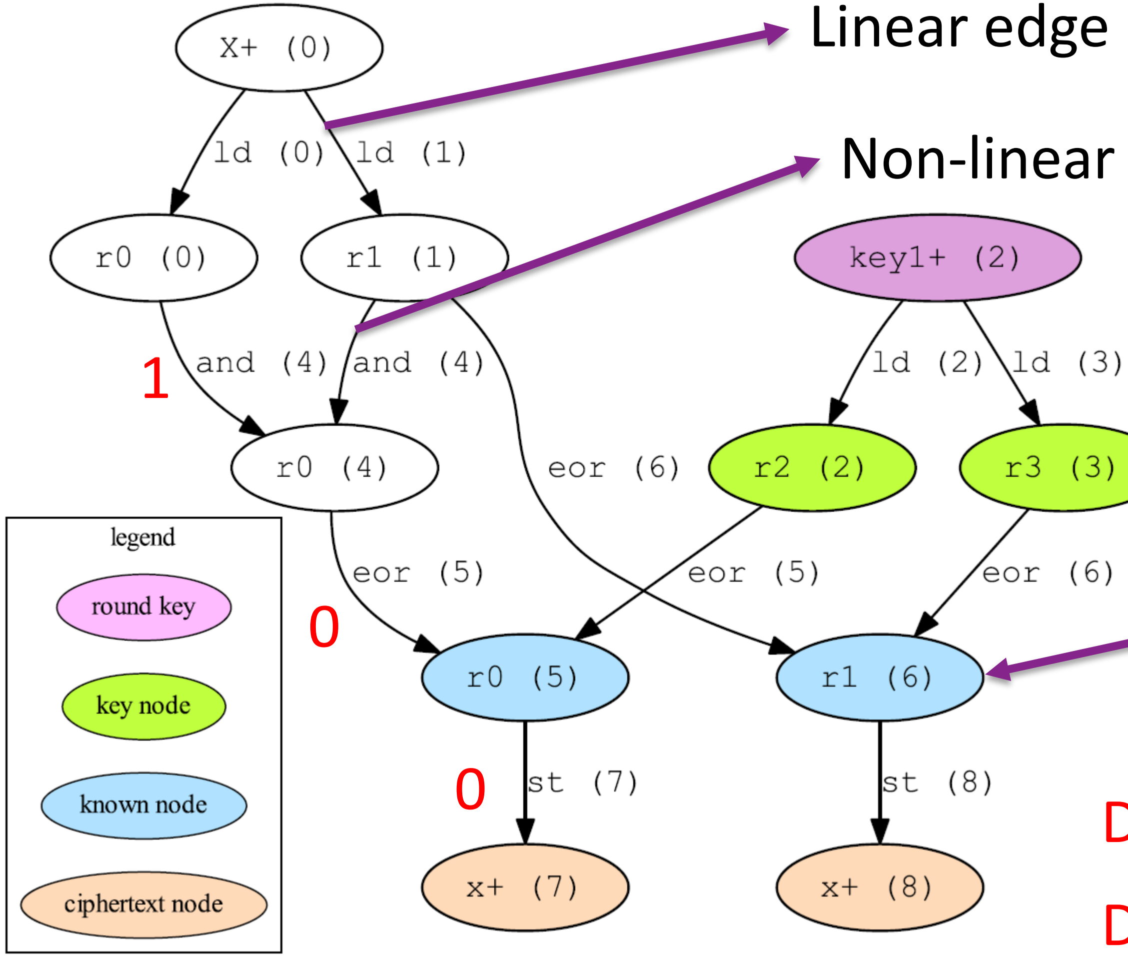


Sample Cipher and DFG Construction

#	Instruction
0	LD r0 X+
1	LD r1 X+
2	LD r2 key1+
3	LD r3 key1+
4	AND r0 r1
5	EOR r0 r2
6	EOR r1 r3
7	ST x+ r0
8	ST x+ r1



Properties of the DFG – Explained



Linear edge

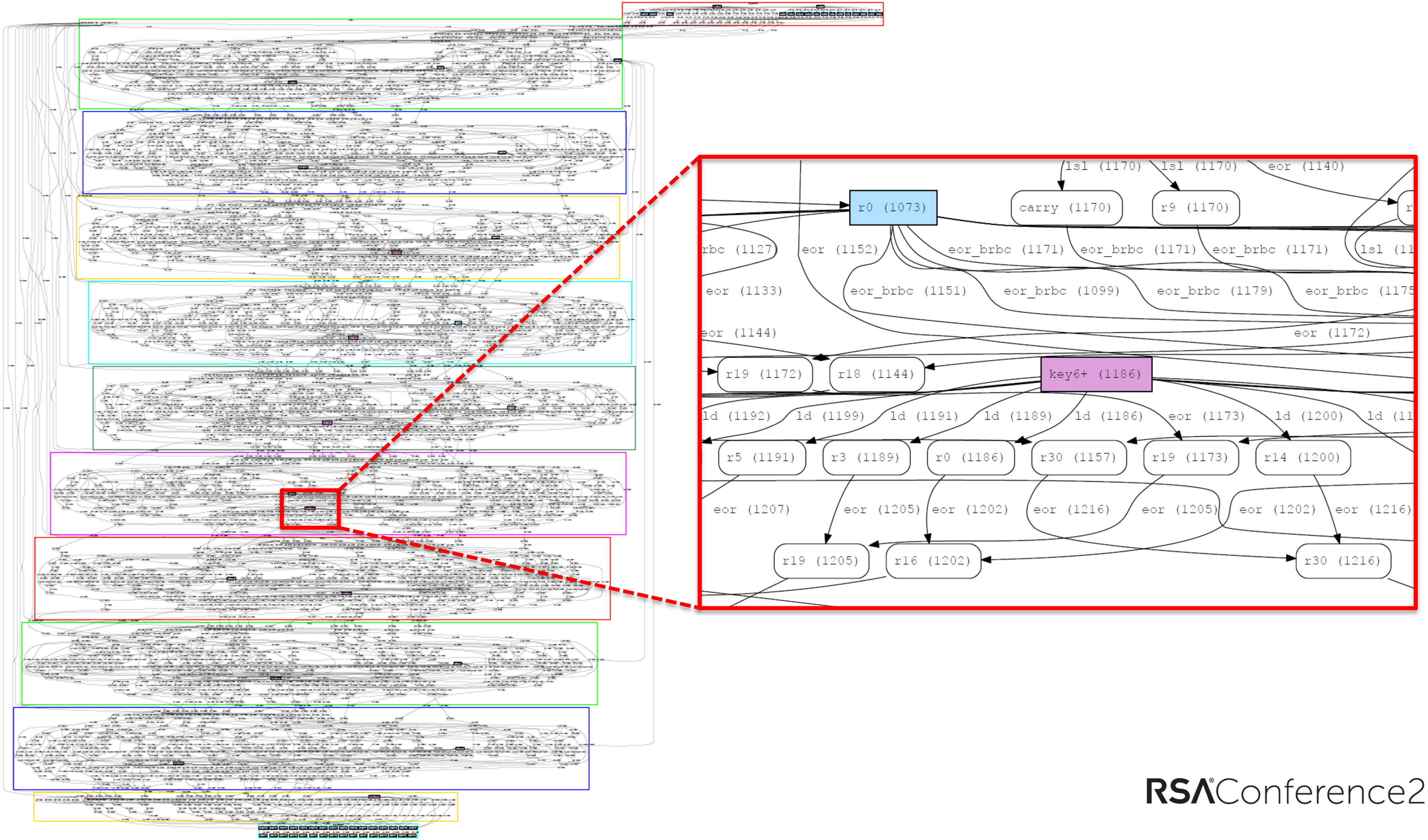
Non-linear edge

Node r3 (3) affects node r1 (6)

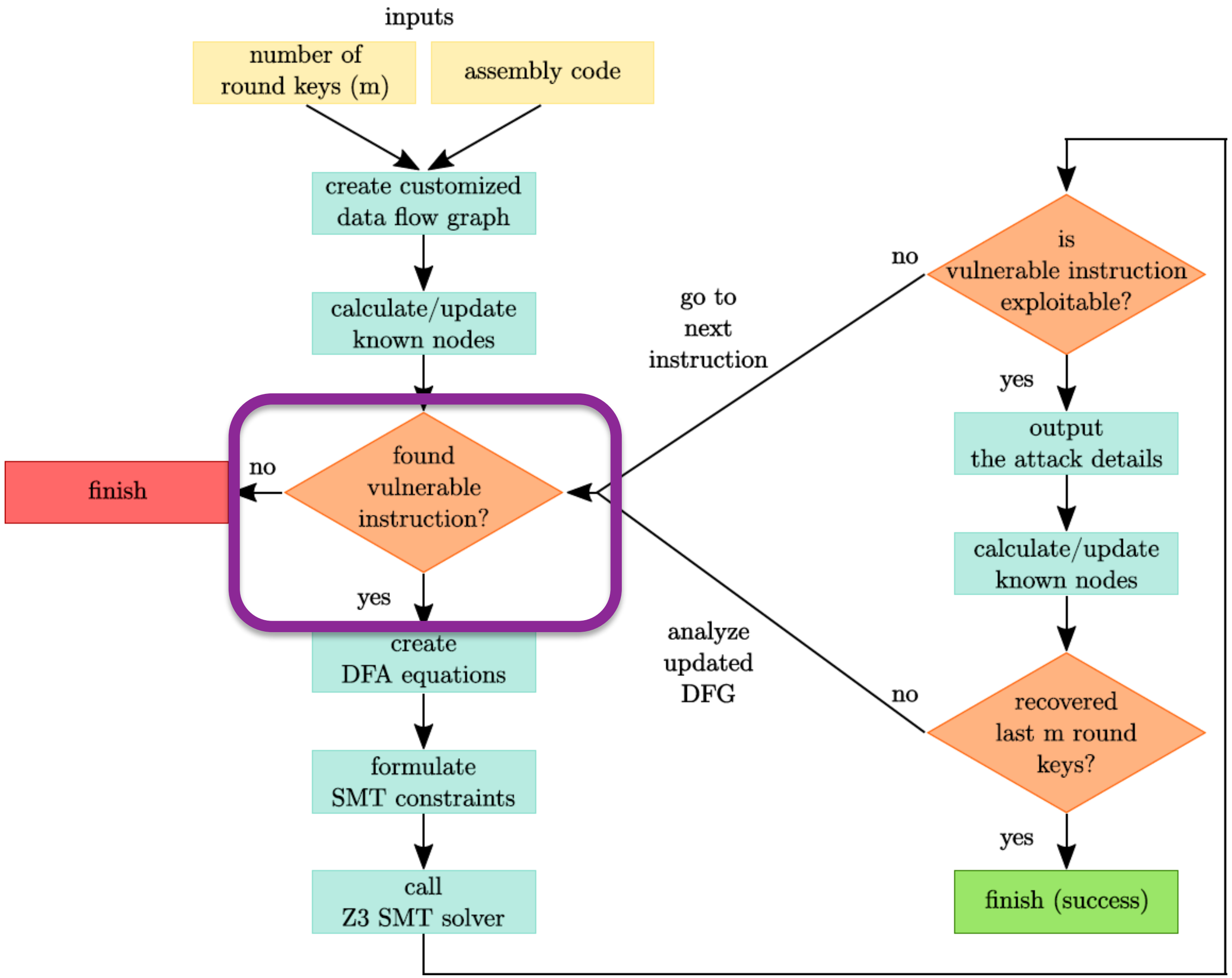
Distance between r0 (0) and r0 (4) is 1

Distance between r0 (0) and x+ (7) is also 1

Real Example – DFG of AES Implementation



TADA – Detailed Process Flow

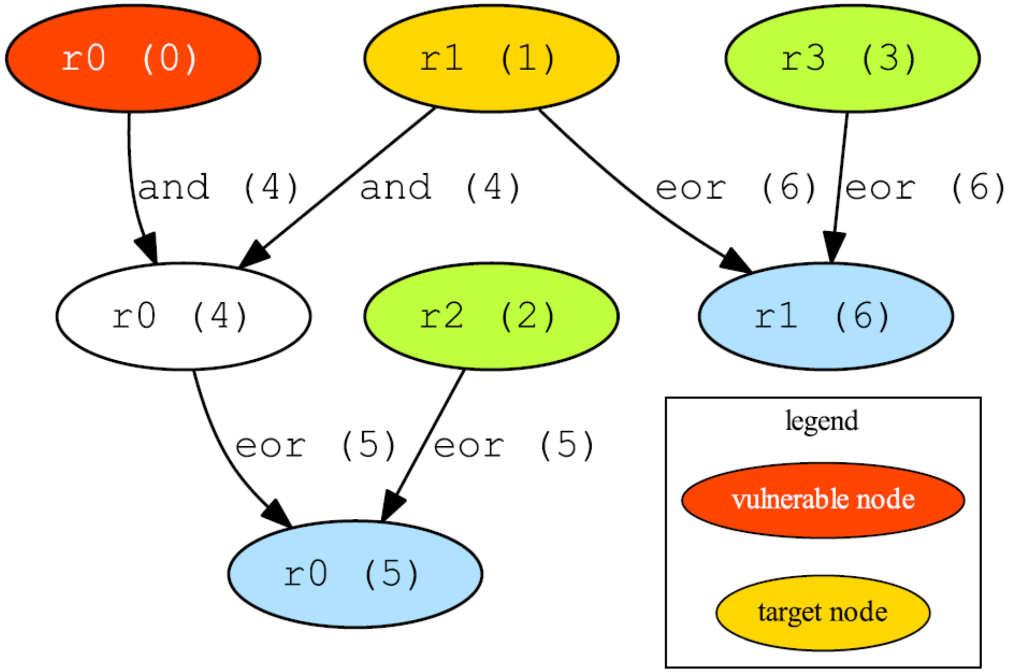


Vulnerable Instructions

- Non-linear
- For a vulnerable instruction, each of its input nodes that is not known can be a *target* node or/and a *vulnerable* node
- A fault will be injected into the *vulnerable* node so that it might reveal information about the *target* node
- TADA creates a subgraph for each pair of target and vulnerable node

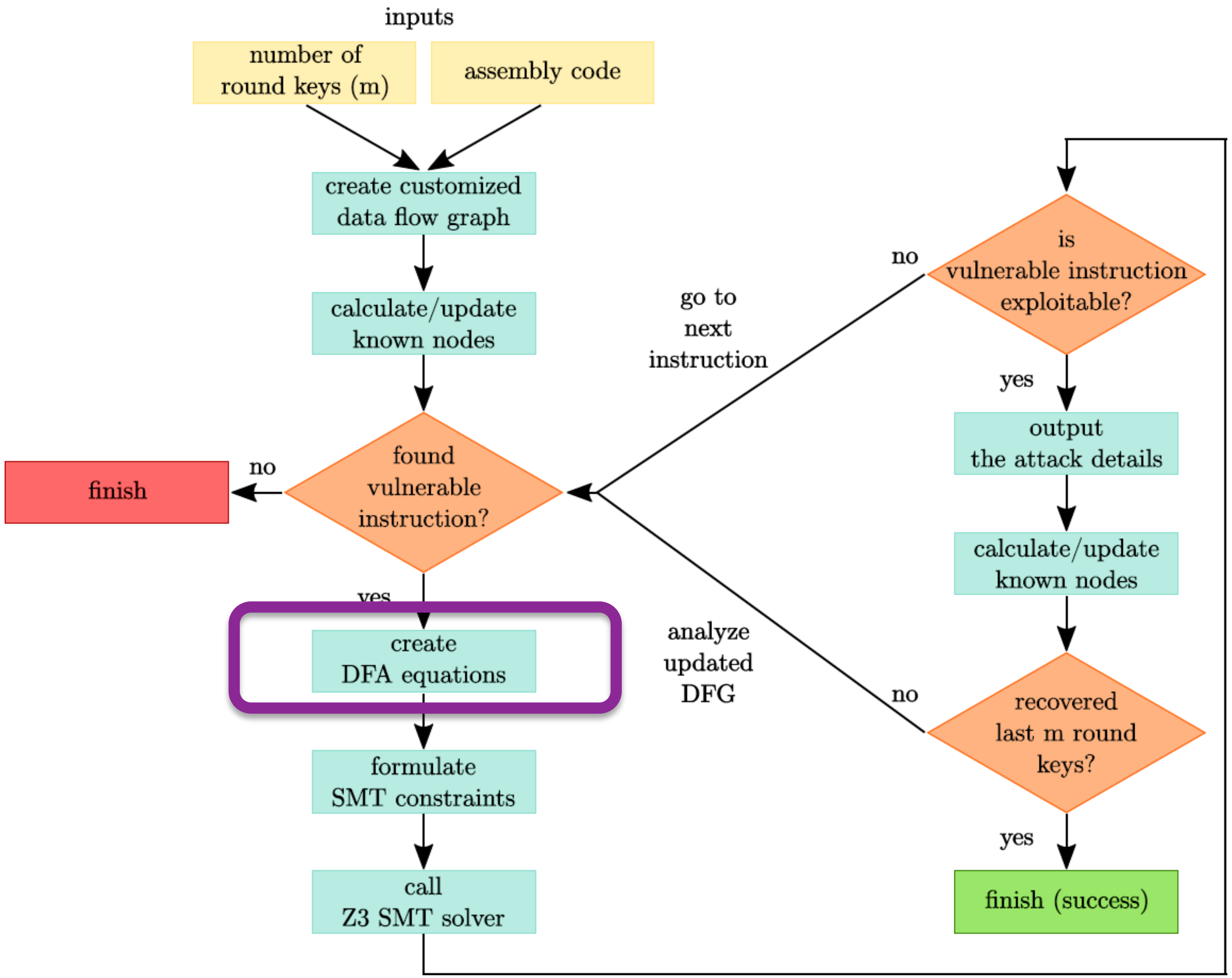
Find Vulnerable Instruction

#	Instruction
0	LD r0 X+
1	LD r1 X+
2	LD r2 key1+
3	LD r3 key1+
4	AND r0 r1
5	EOR r0 r2
6	EOR r1 r3
7	ST x+ r0
8	ST x+ r1



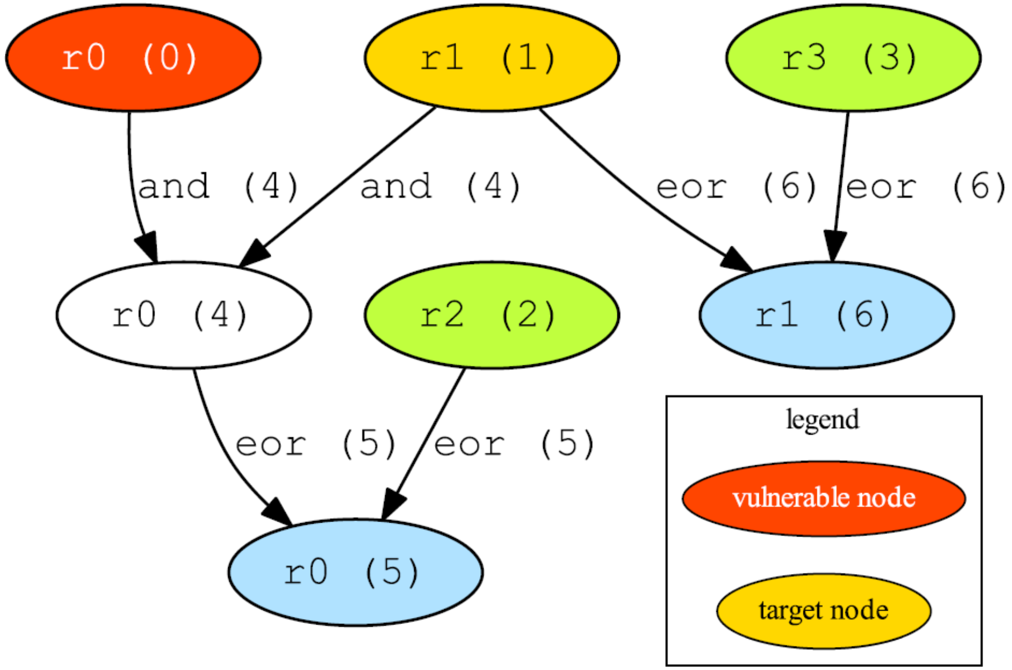
Recall that r2 (2) and r3 (3) are the key nodes

TADA – Detailed Process Flow



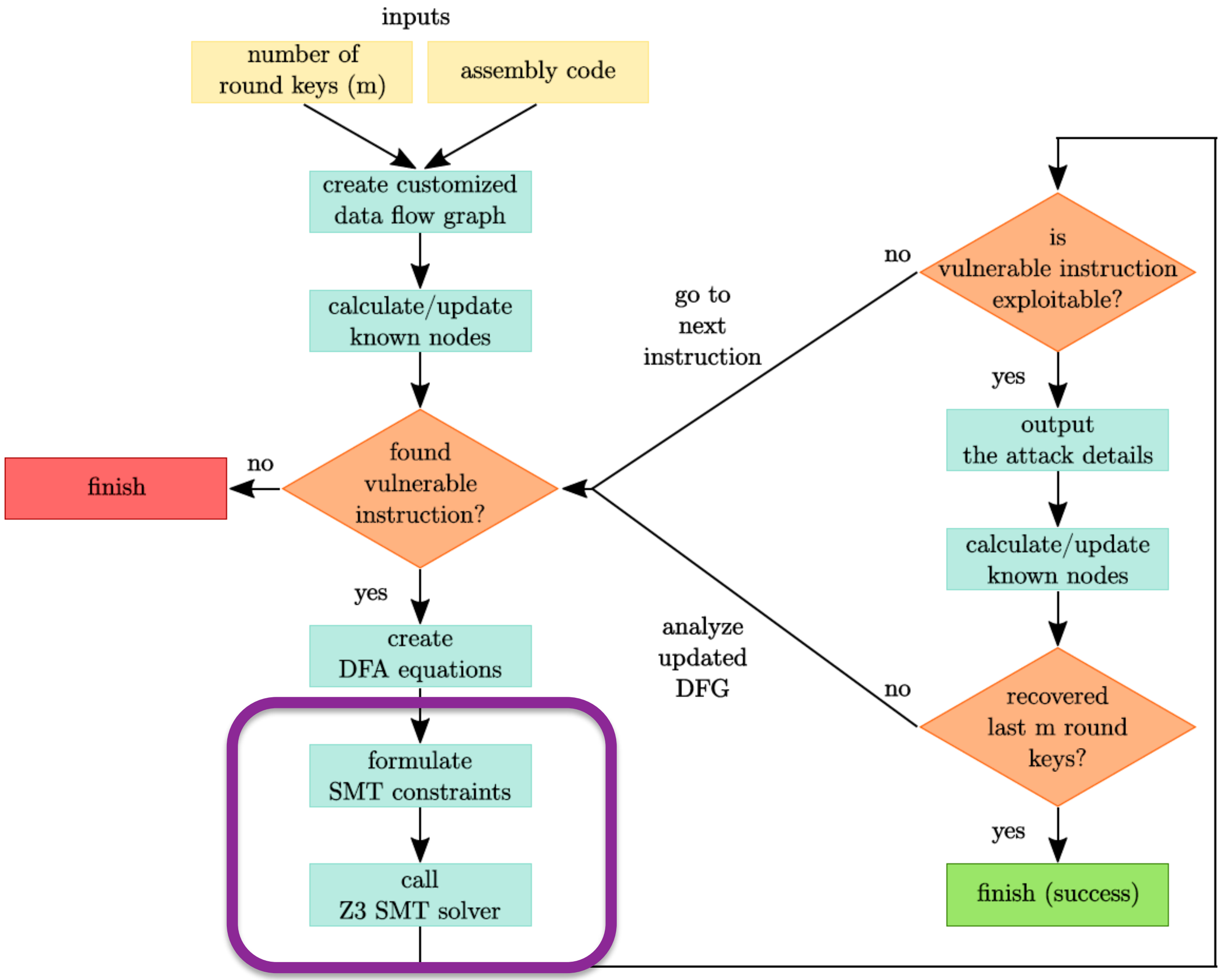
Create DFA Equations

Correct execution	Faulted execution	Fault mask
$(a)\ \ r0(4) = r0(0)\ \ \&\ \ r1(1)$ $(b)\ \ r0(5) = r0(4) \oplus r2(2)$ $(c)\ \ r1(6) = r1(1) \oplus r3(3)$	$(d)\ \ r0(4)' = r0(0)'\ \ \&\ \ r1(1)$ $(e)\ \ r0(5)' = r0(4)' \oplus r2(2)$	$r0(0)' = r0(0) \oplus \delta$

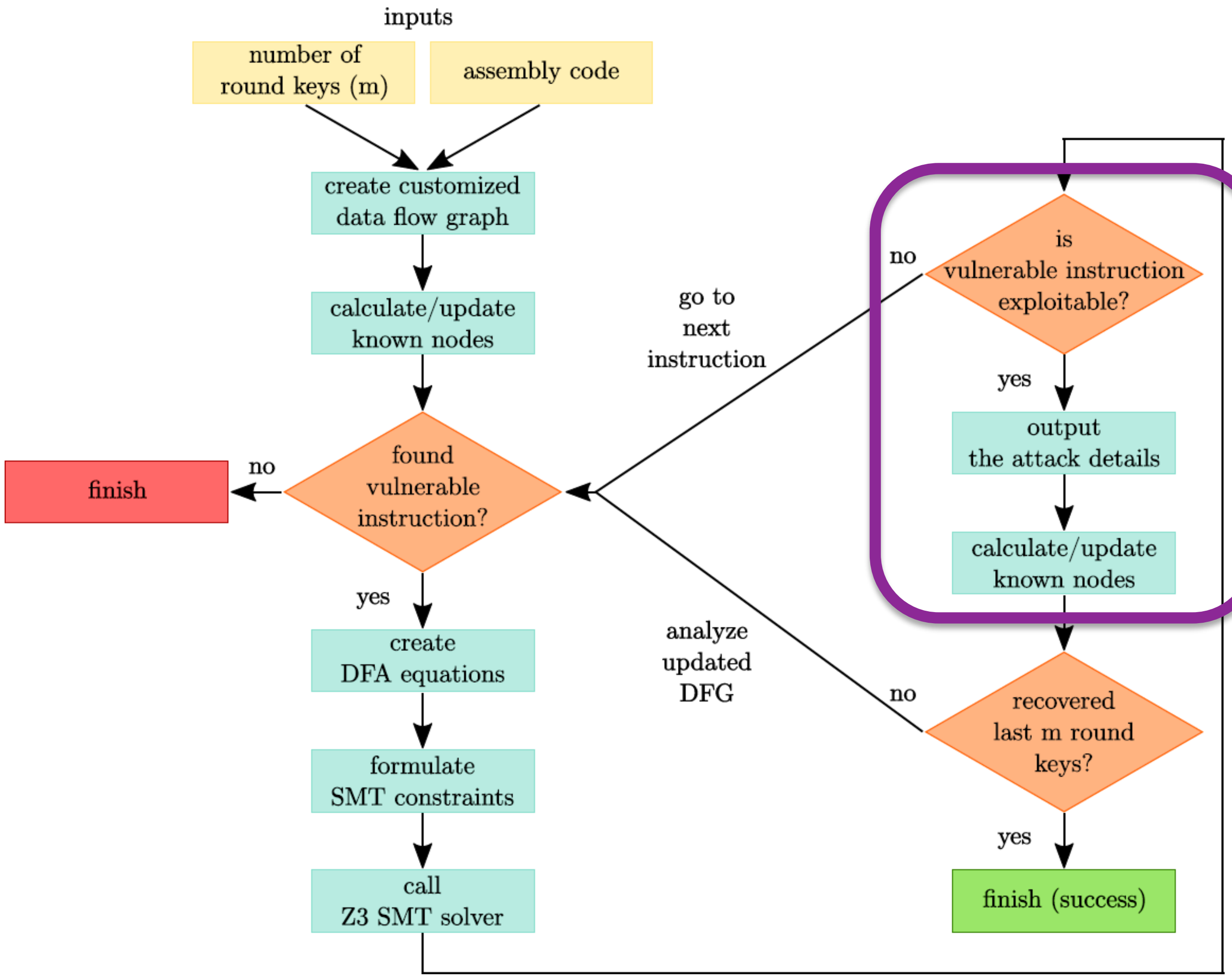


RSAConference2019

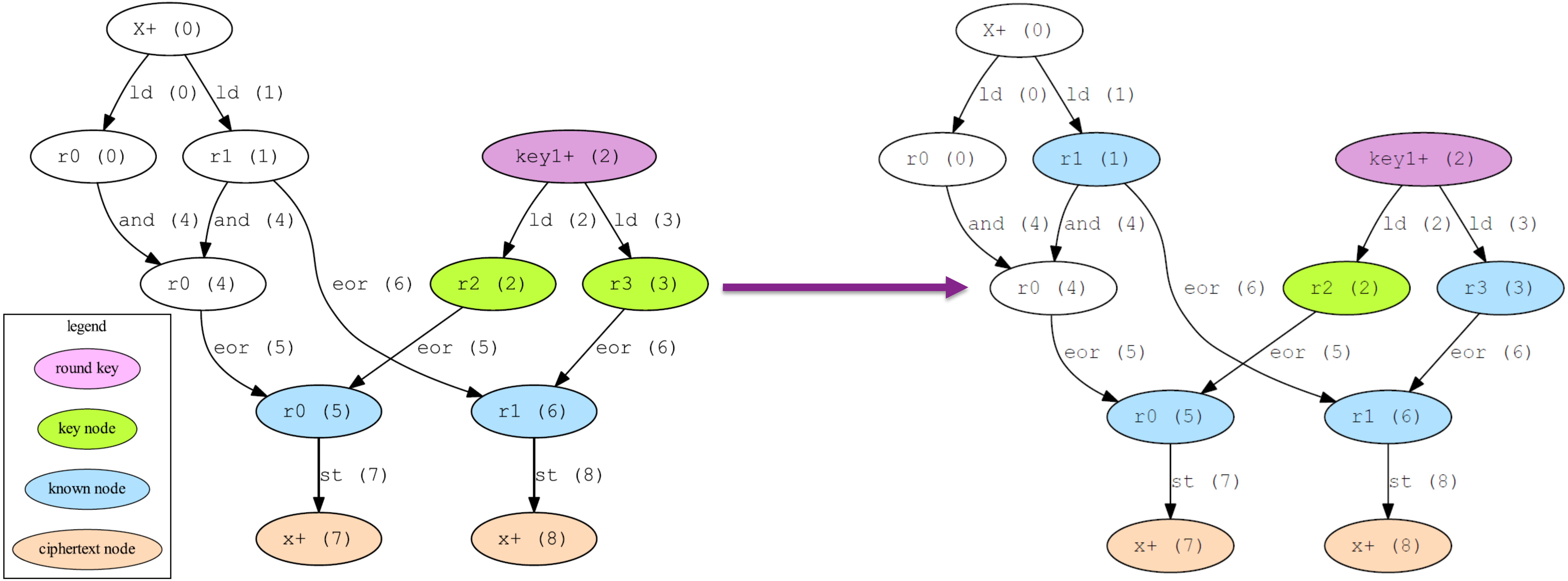
TADA – Detailed Process Flow



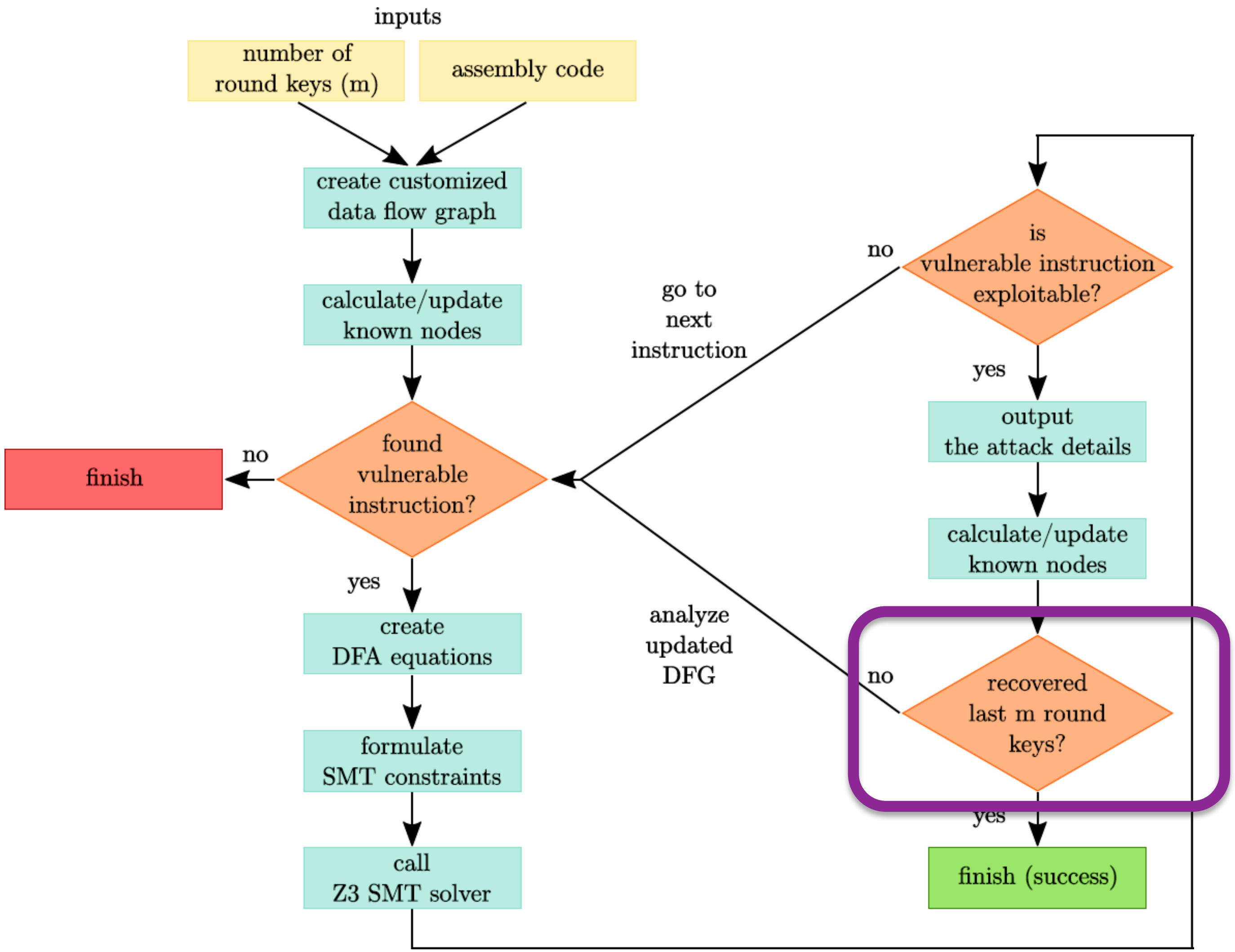
TADA – Detailed Process Flow



Update Known Nodes

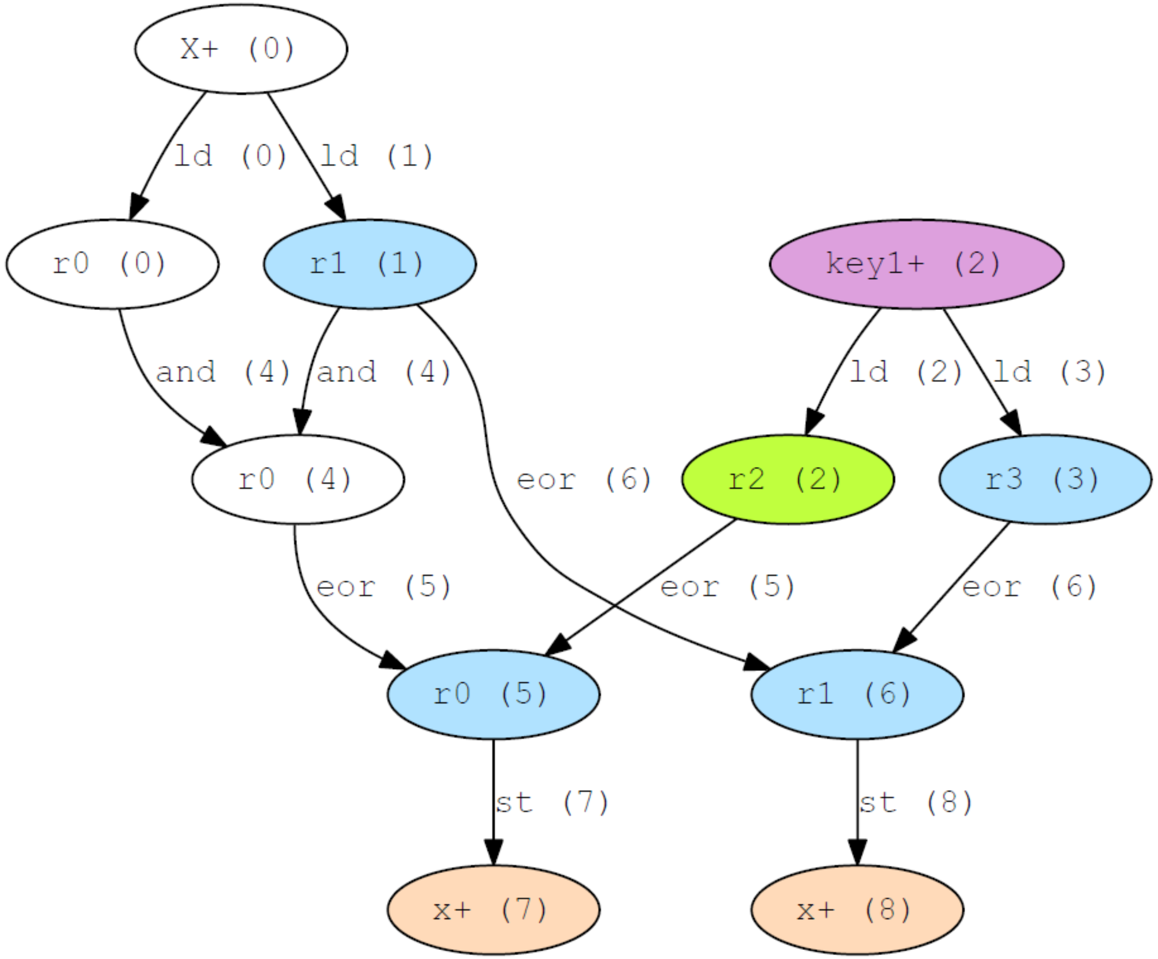


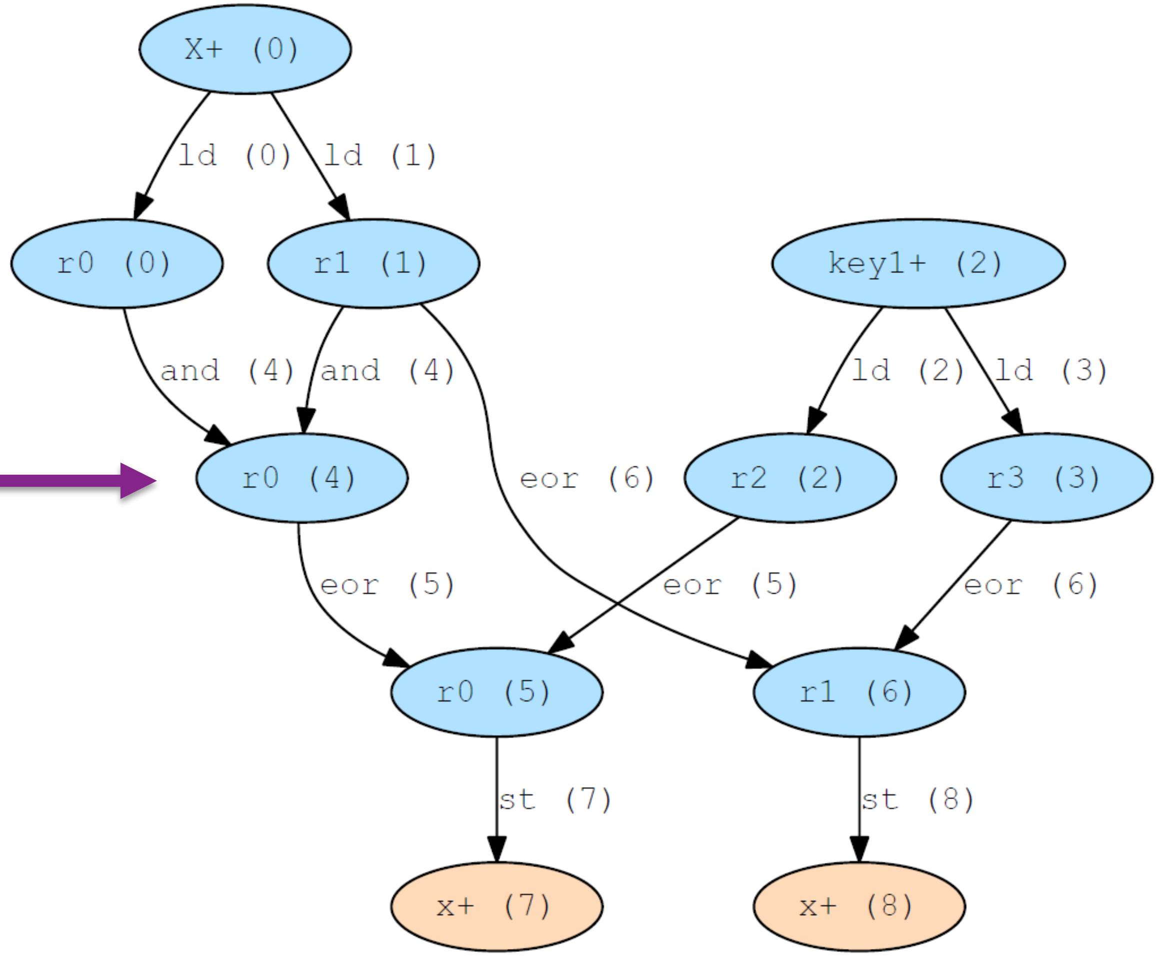
TADA – Detailed Process Flow



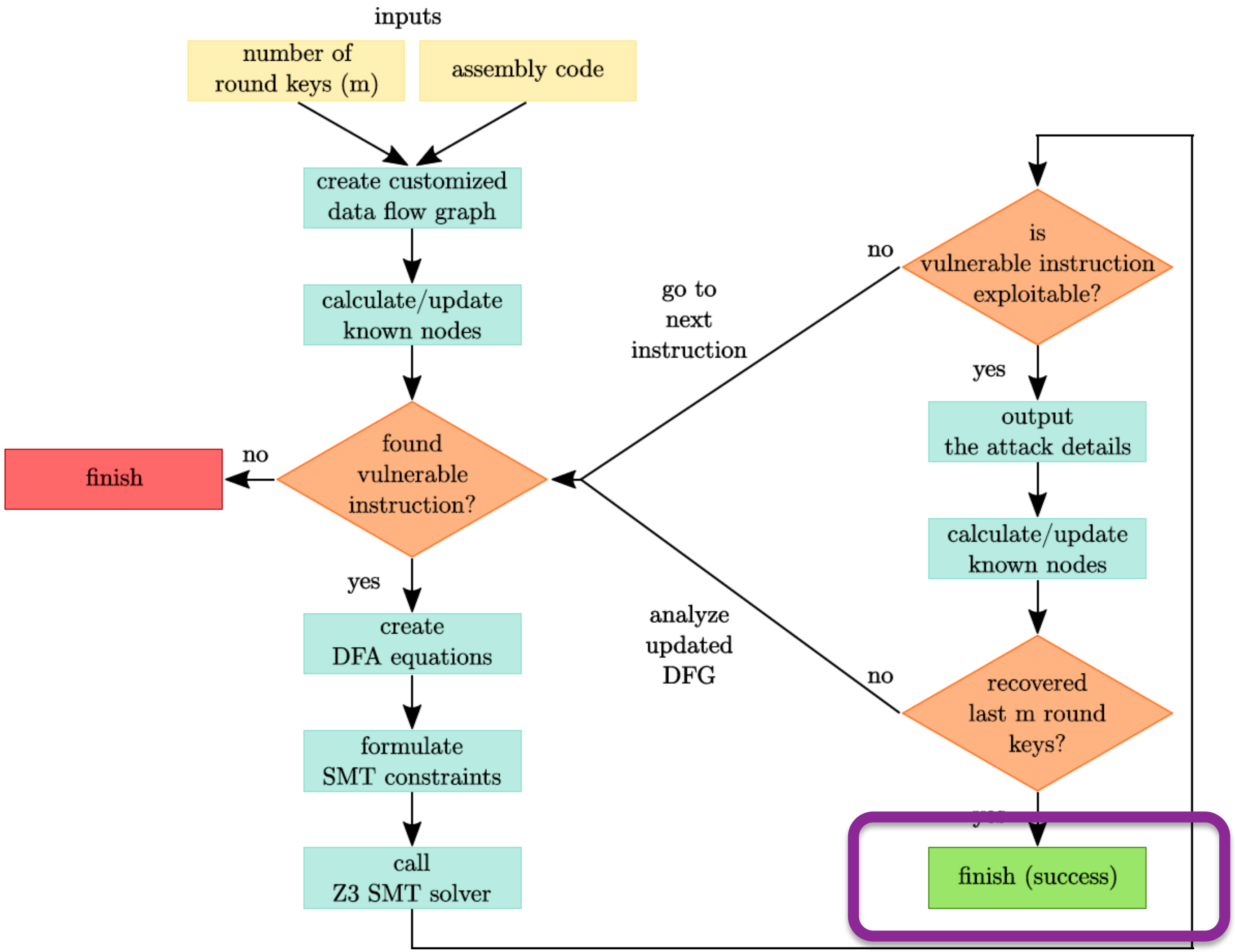
Not yet!

One More Iteration





TADA – Detailed Process Flow



RSAConference2019

Evaluation Results

Cipher implementation	SIMON	SPECK	AES	PRIDE
# of lines of code (unrolled)	1,272	663	2,057	1590
# of nodes in DFG	1,595	843	2,060	1763
# of edges in DFG	2,709	1,562	3,209	2586
evaluation time (min)	17.2	9.8	298.7	4.6
fault attack found	[TBM14]	**new**	[Gir05]	**new**
# of known nodes before attack	66	32	69	16
# of known nodes after attack	162	117	149	196
# of round keys found	2	2	1	2

[TBM14] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Differential fault analysis on the families of Simon and Speck ciphers. FDTC 2014.
[Gir05] Christophe Giraud. DFA on AES. Conference on AES 2005.

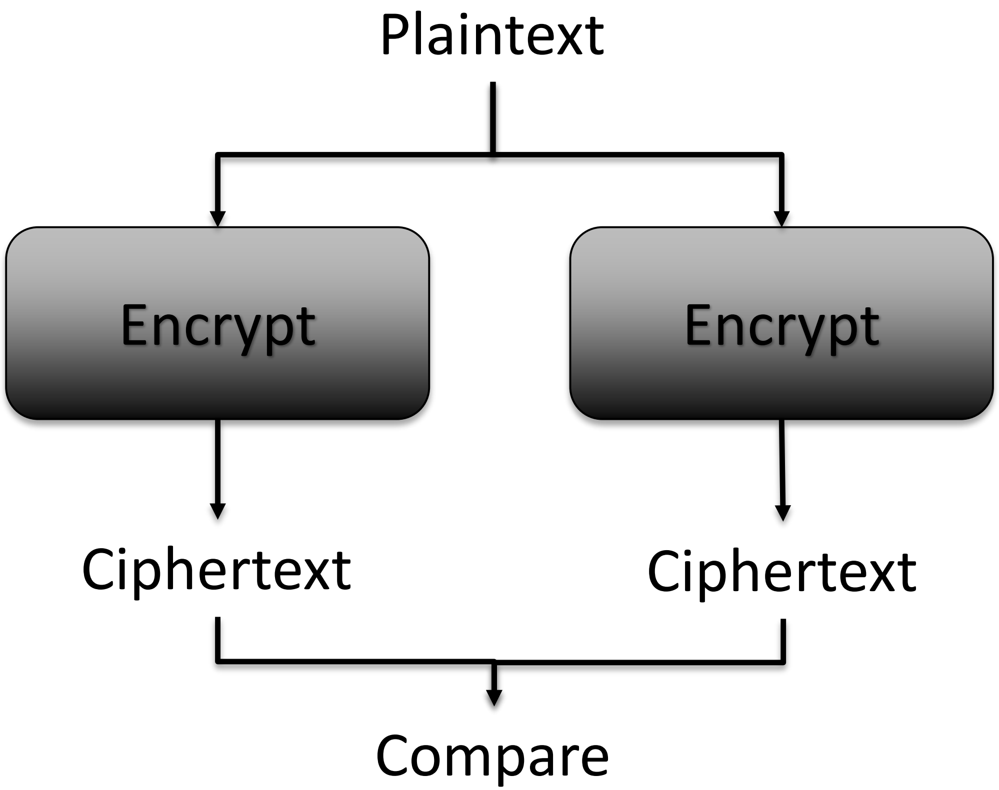
RSAConference2019

Countermeasures

How many rounds to protect?

Standard Duplication/Triplication Countermeasure

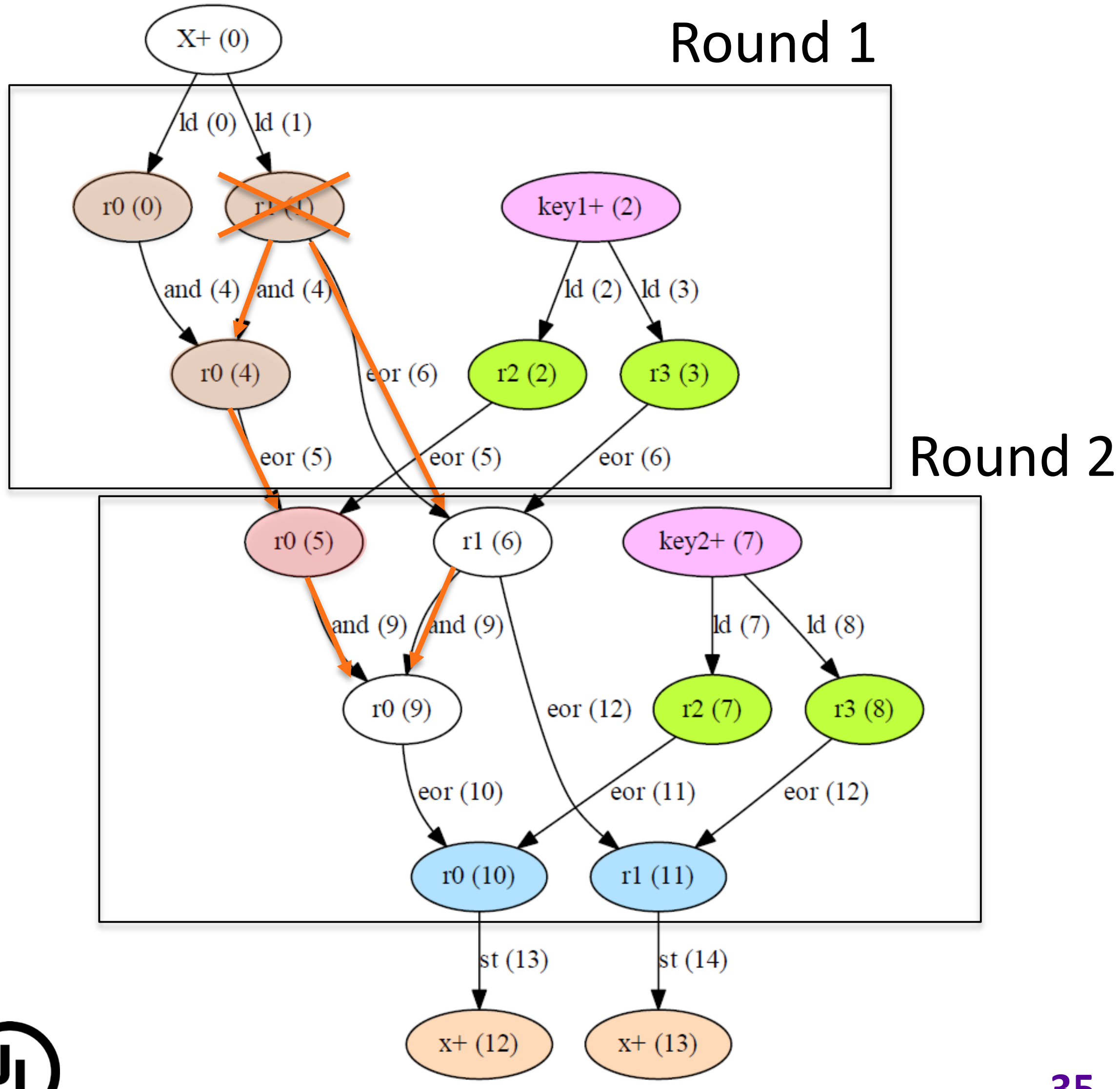
- Popular in industrial applications
- Either area or time redundancy
- Expensive overheads
- Resources can be saved in case it is not necessary to protect the entire cipher



Countermeasure implementation based on TADA

- We know which nodes are provably exploitable by TADA
- We are now trying to find the *earliest* node possible to affect the target node, such that there are no collisions
- This information will tell us what is the earliest round where the fault can be injected

Back to the Example – with 2 rounds



Target node	Vulnerable node
r0 (5)	r1 (6)
r1 (6)	r0 (5)

How can we attack r0 (5)?

- r0 (4)
- r0 (0)
- ~~r1 (1)~~ →collision

As a result, we have extended the attack to the second last round

RSAConference2019

How Many Rounds to Protect?

Cipher implementation	SIMON	SPECK	AES	PRIDE
Earliest round attacked	$R-2$	$R-3$	$R-3$	$R-3$

- Resources for countermeasures can be saved as follows:
  - SIMON – over 90% (3 out of 32 rounds)
  - SPECK – over 81% (4 out of 22 rounds)
  - AES – over 60% (4 out of 10 rounds)
  - PRIDE – over 80% (4 out of 20 rounds)

RSAConference2019

Summary

Short Recap

- All the block ciphers have been shown to be vulnerable against Differential Fault Analysis
- Automated methods can help to accurately find vulnerabilities in implementations without the need of human intervention
- Application of countermeasures can be iteratively tested until the implementation is secure

Apply It

- Next week you should:
  - Identify embedded block cipher implementations that are deployed in the field and are susceptible to fault injection attacks (e.g. in IoT devices)
- In the first three months following this presentation you should:
  - Being able to automatically analyze these implementations
- Within six months you should:
  - Have a policy for applying automated analysis for every new block cipher implementation

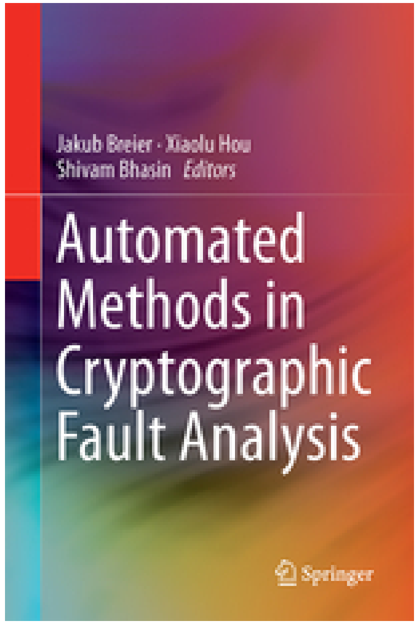
Resources

- X. Hou, J. Breier, F. Zhang and Y. Liu. Fully Automated Differential Fault Analysis on Software Implementations of Cryptographic Algorithms. Cryptology ePrint Archive: Report 2018/545 (https://eprint.iacr.org/2018/545).
- J. Breier, X. Hou and Y. Liu. Fault Attacks Made Easy: Differential Fault Analysis Automation on Assembly Code. Cryptology ePrint Archive: Report 2017/829 (https://eprint.iacr.org/2017/829). Published in TCHES 2018 Issue 2, IACR.

- Future works: http://jbreier.com/research.html

Book on the Topic

- J. Breier, X. Hou, S. Bhasin (eds.): Automated Methods in Cryptographic Fault Analysis, Springer, 2019 (coming in April).



Offers a complete perspective on protecting block ciphers against fault attacks – from analysis to deployment

RSA Conference 2019

Thanks for attention!

Any questions?

Jakub Breier

Underwriters Laboratories, Singapore

jbreier@jbreier.com

http://jbreier.com