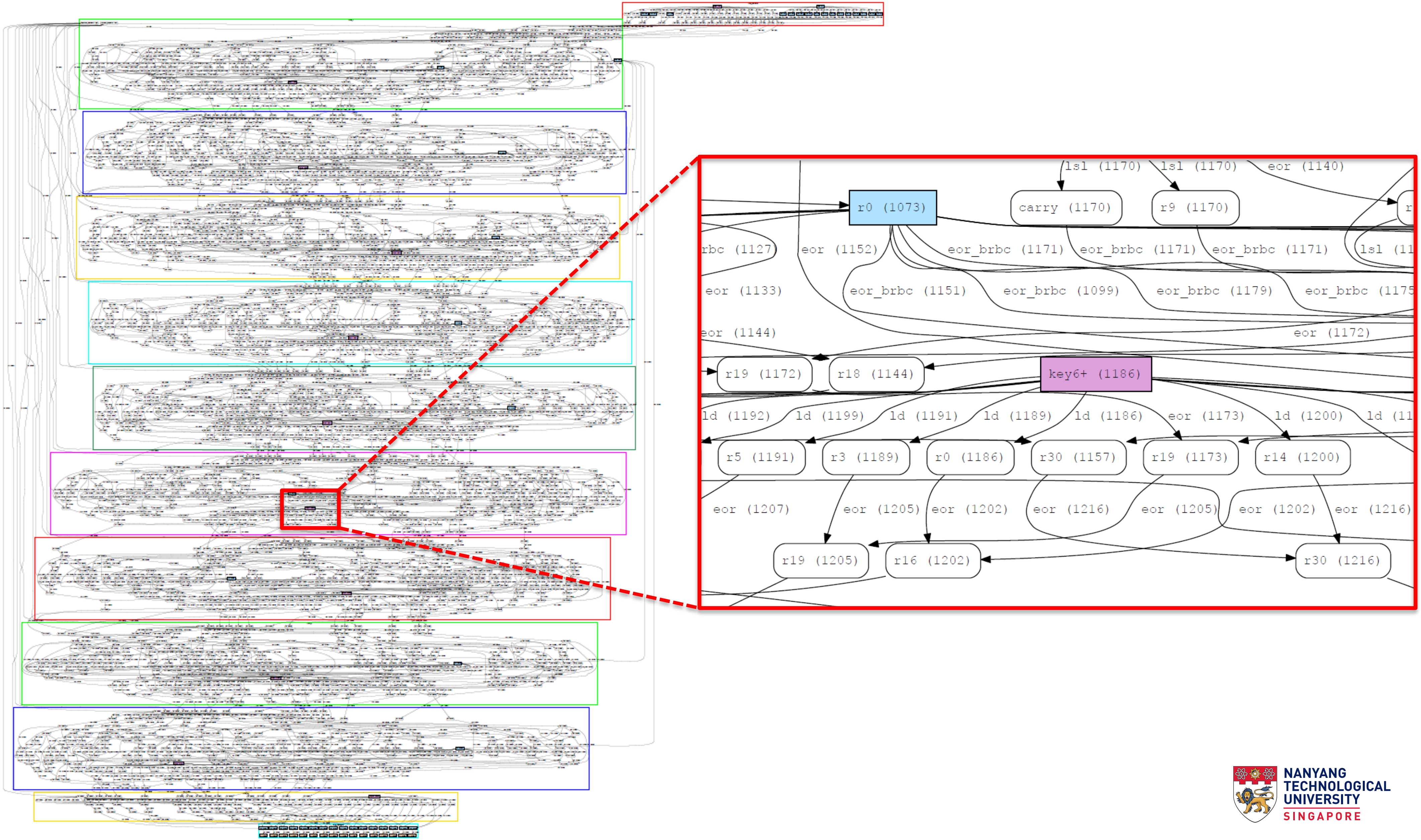
Fully Automated Differential Fault Analysis on Software Implementations of Block Ciphers

Xiaolu Hou[1], Jakub Breier[2], Fuyuan Zhang[3], and Yang Liu[2]

[1] National University of Singapore, Singapore
[2] HP-NTU Digital Manufacturing Corporate Lab, Singapore
[3] Max Planck Institute, Karlsruhe, Germany

CHES'19, 28 Aug 2019

Data Flow Graph of Software Implementation of AES



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Our Contribution

- We developed a method that works on assembly implementations of block ciphers, it identifies spots vulnerable to differential fault analysis (DFA) by bit flips, and verifies whether those spots are exploitable
- Our method is *sound* – if it marks the spot as exploitable, it is provably exploitable
  - The prototype tool outputs the identified attack
- Furthermore, we developed a way to check how many rounds should be protected by a countermeasure to be able to avoid DFA to vulnerable spots

Tool for Automated DFA on Assembly

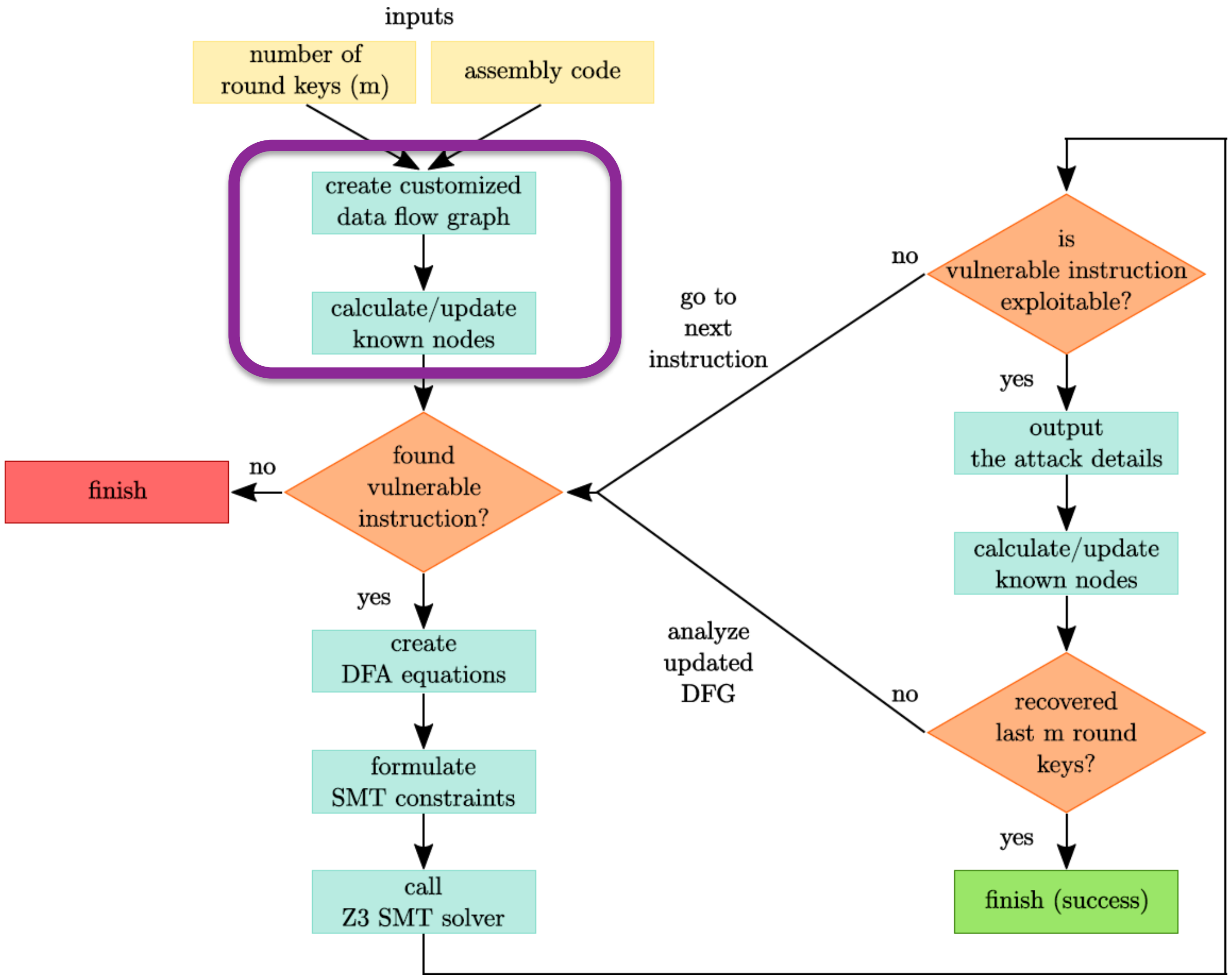
Tool for Automated DFA on Assembly – TADA

- 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

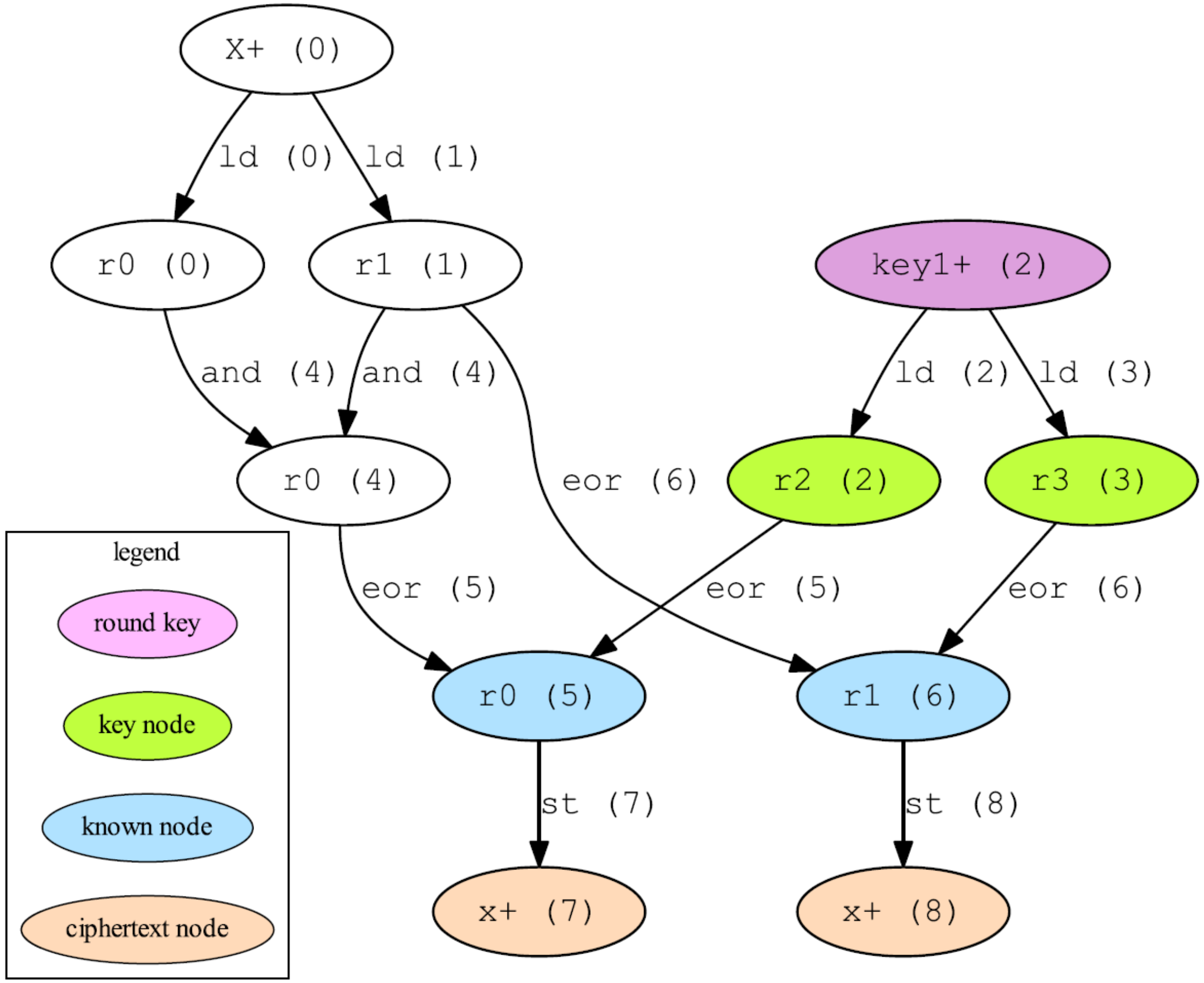
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

TADA – Detailed Process Flow

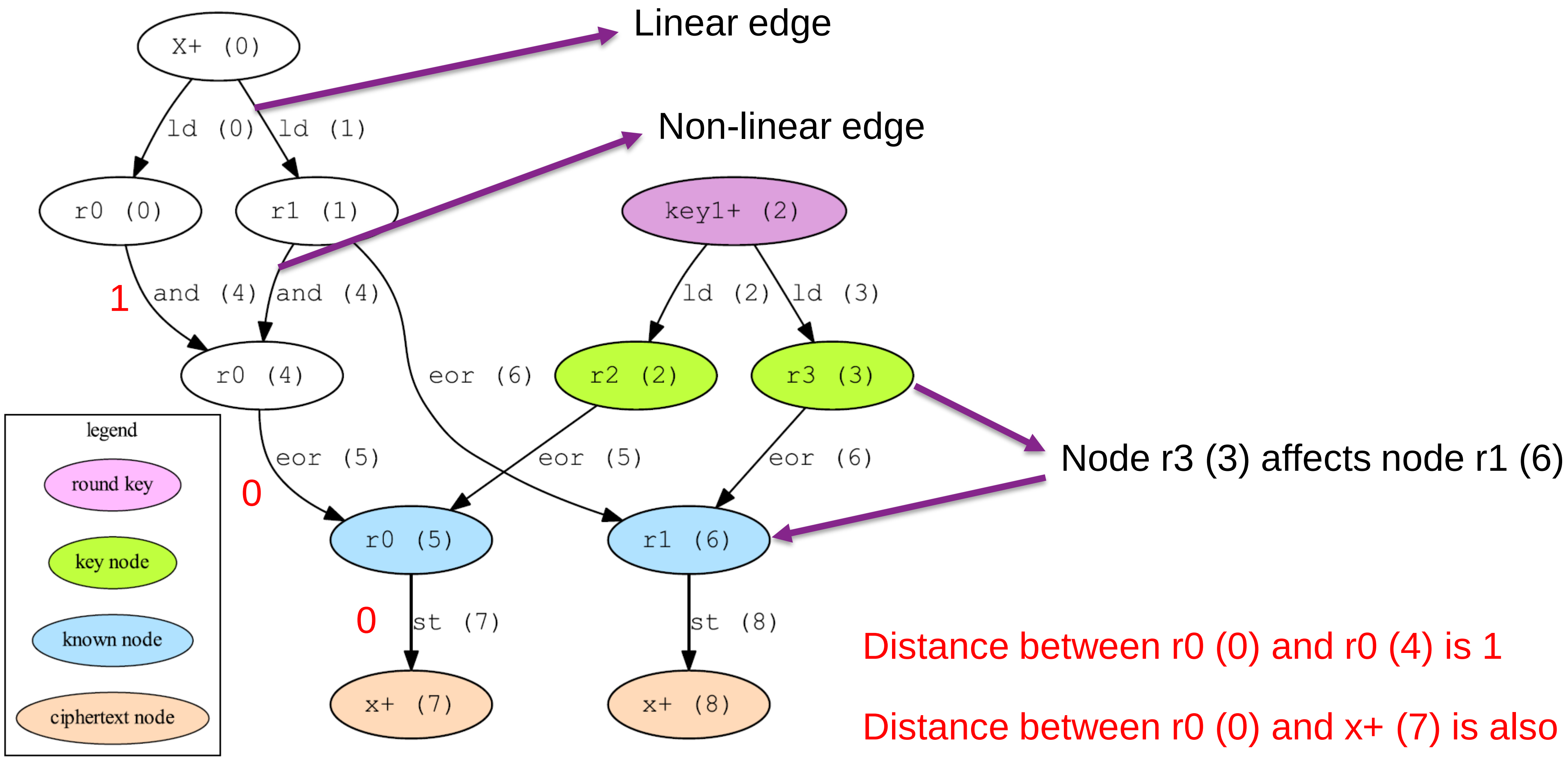


Sample Cipher and DFG Construction

#	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



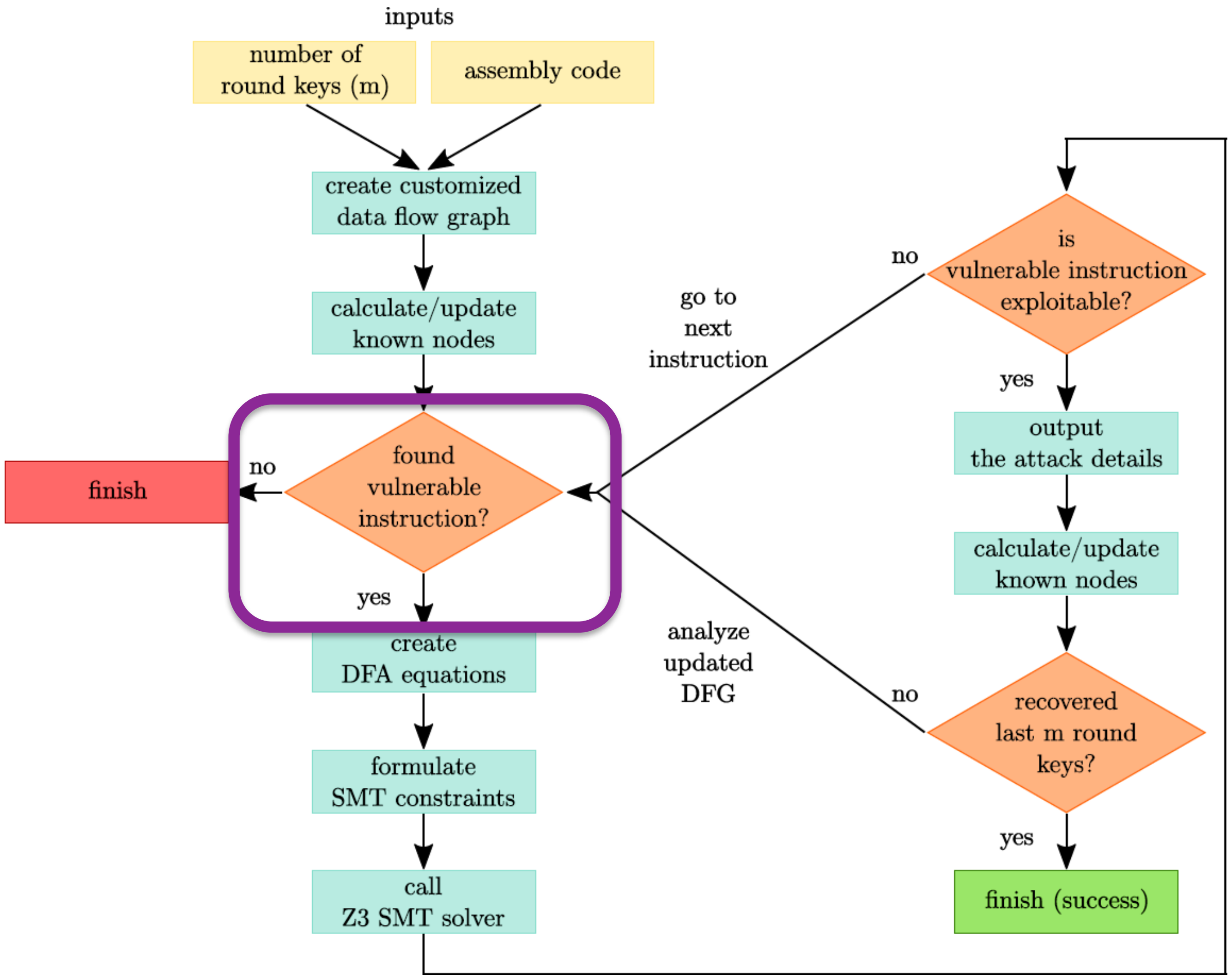
Properties of the DFG – Explained



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

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

TADA – Detailed Process Flow



Vulnerable Instructions

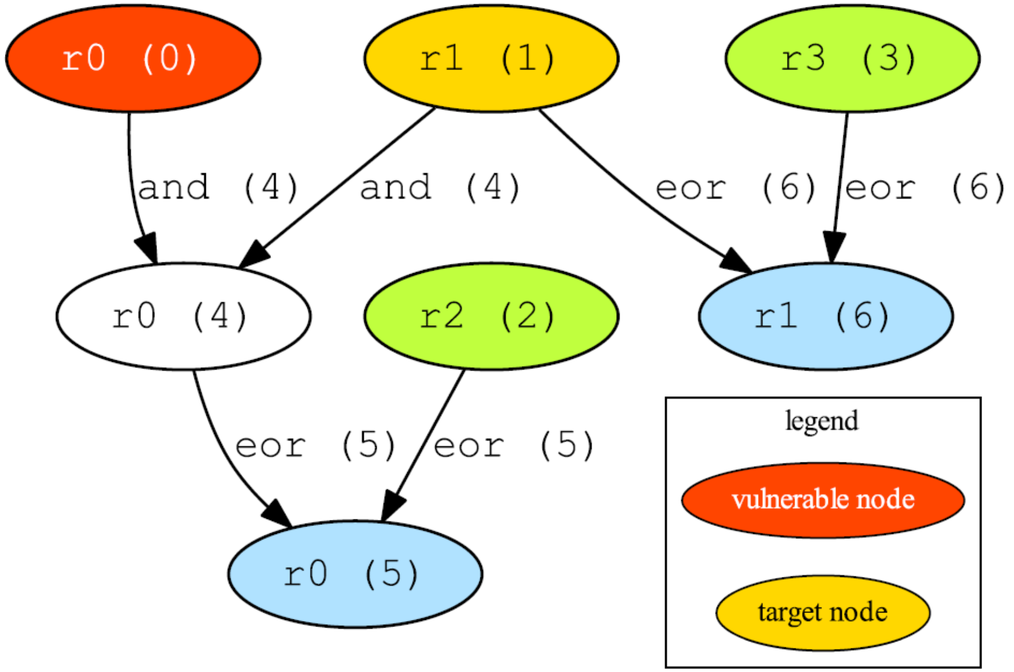
- For a vulnerable instruction, each of its input nodes that is not known can be a *target* node or/and a *vulnerable* node

- 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

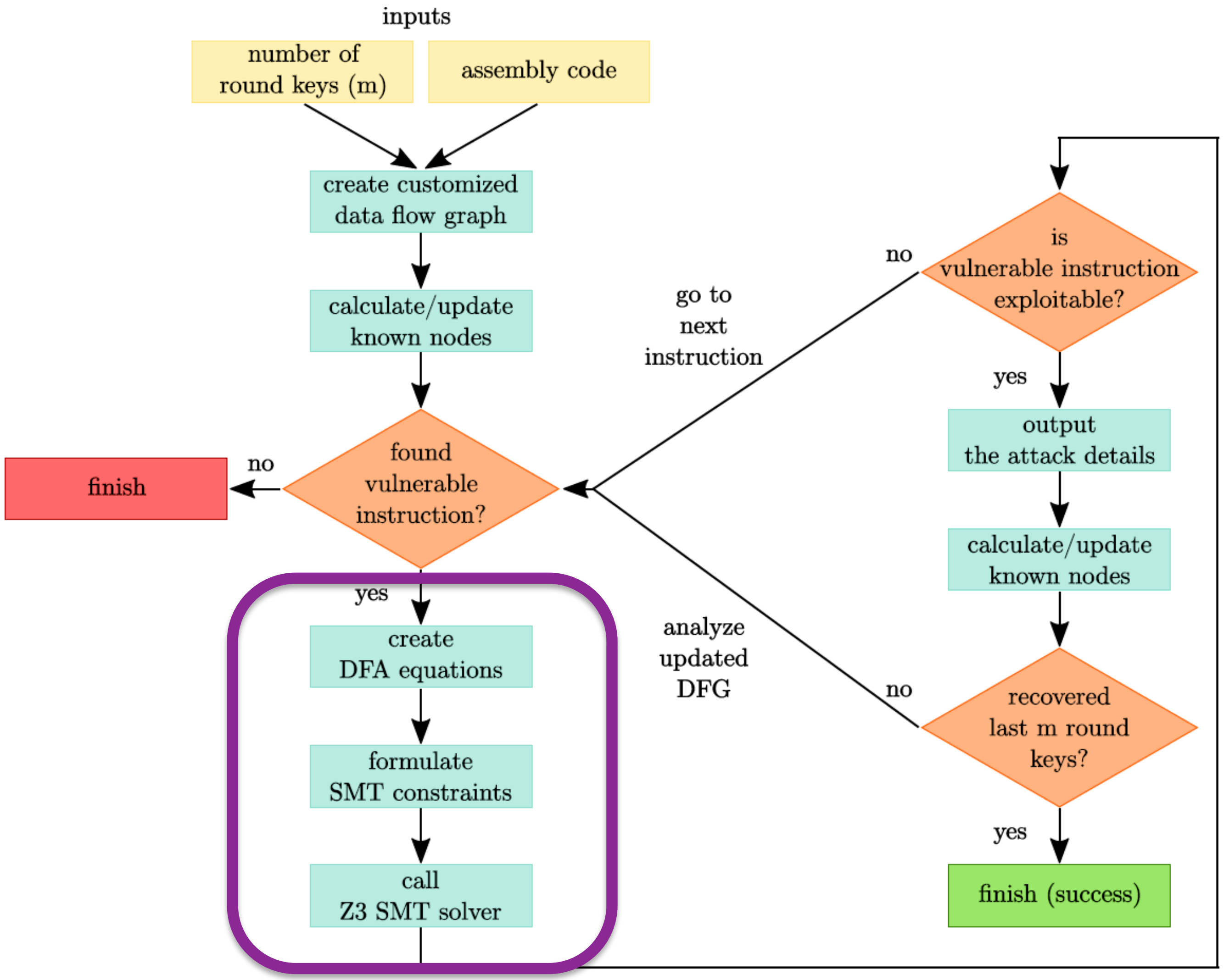
Find Vulnerable Instruction

#	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

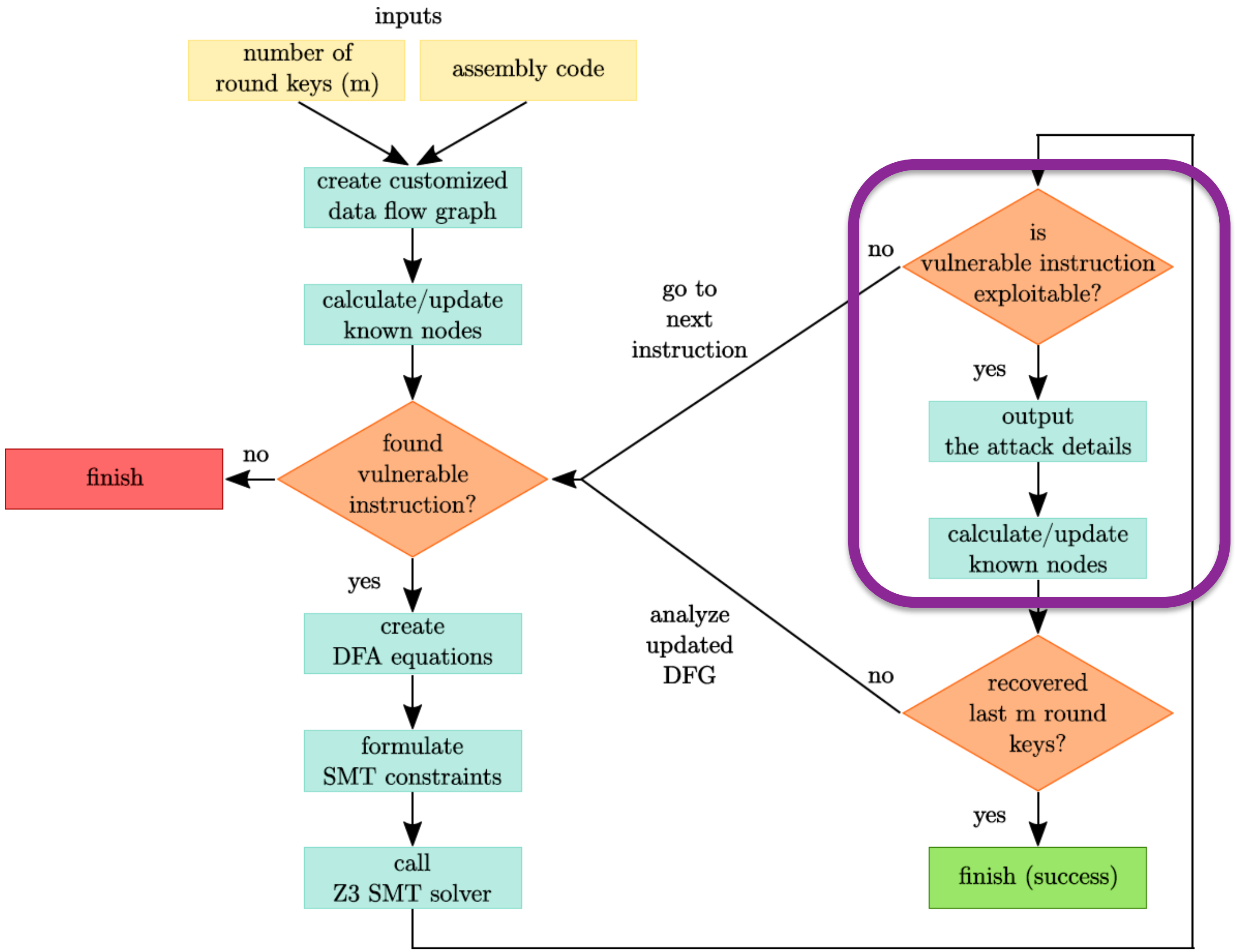


Recall that r2 (2) and r3 (3) are the key nodes

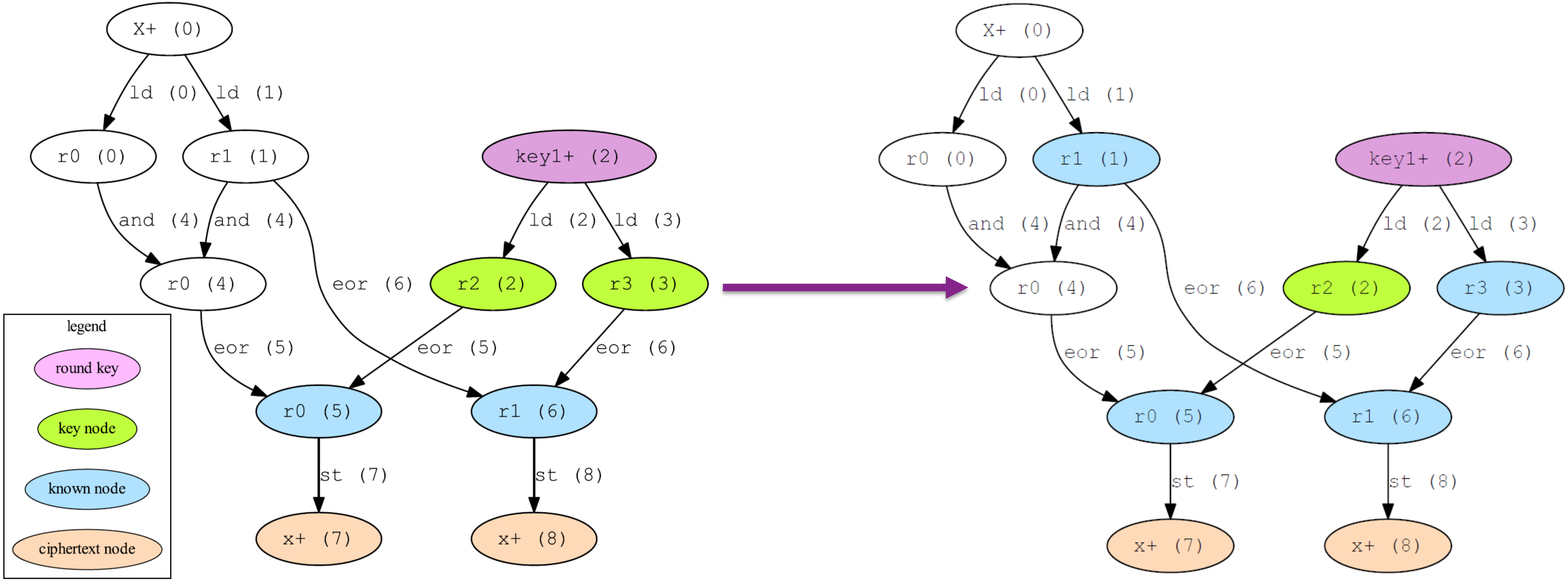
TADA – Detailed Process Flow



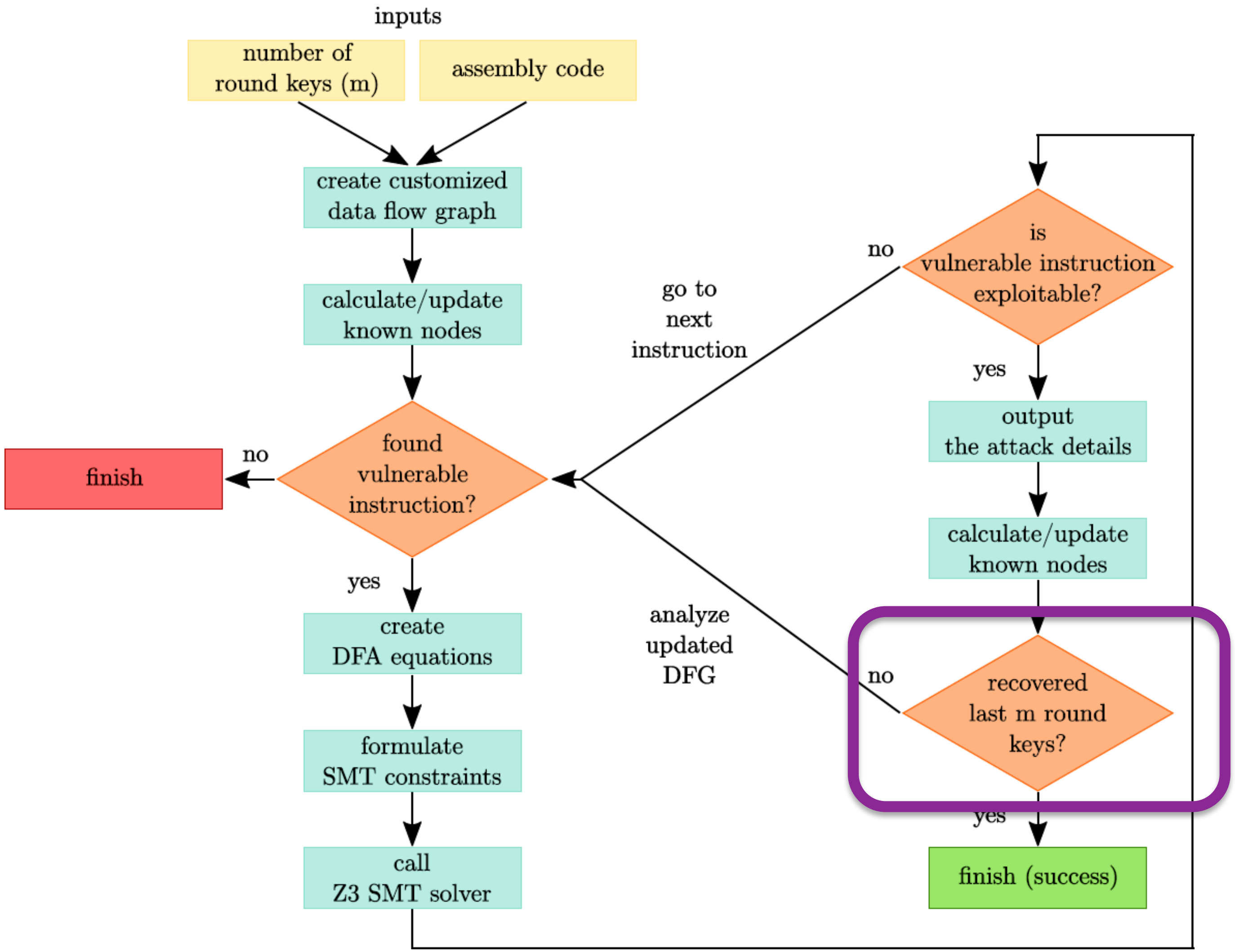
TADA – Detailed Process Flow



Update Known Nodes

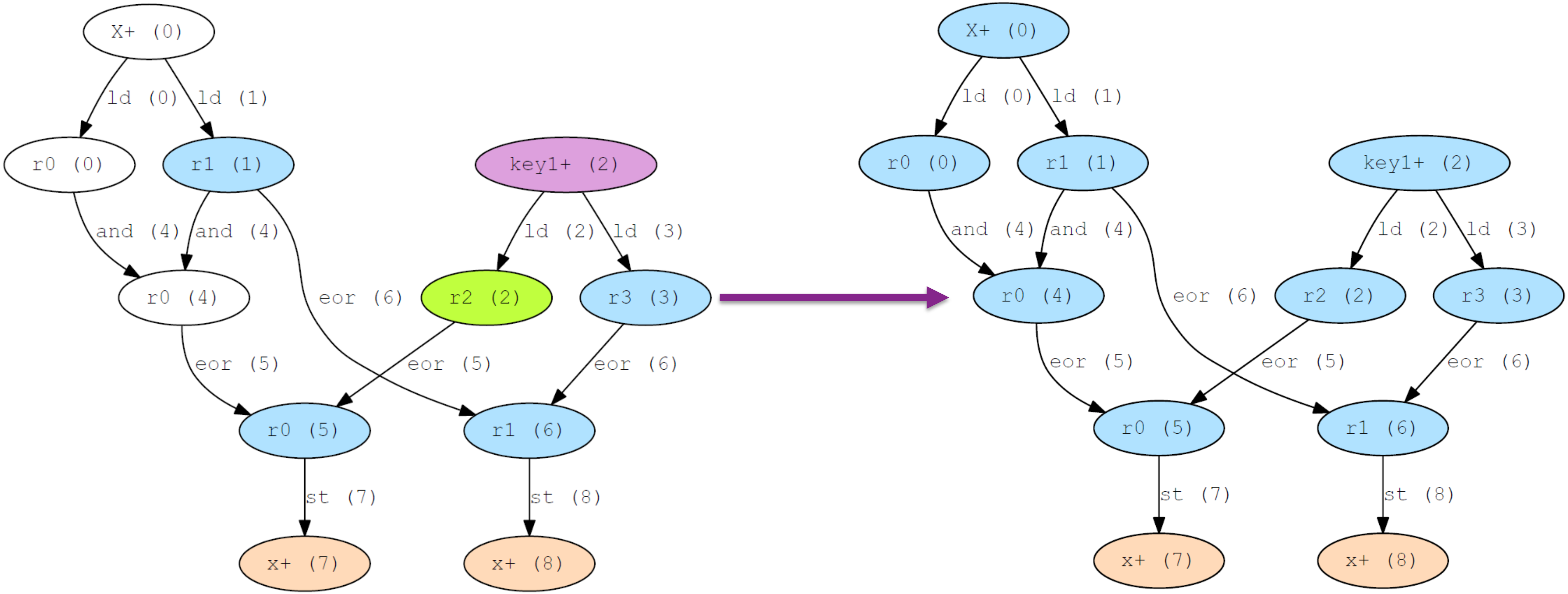


TADA – Detailed Process Flow

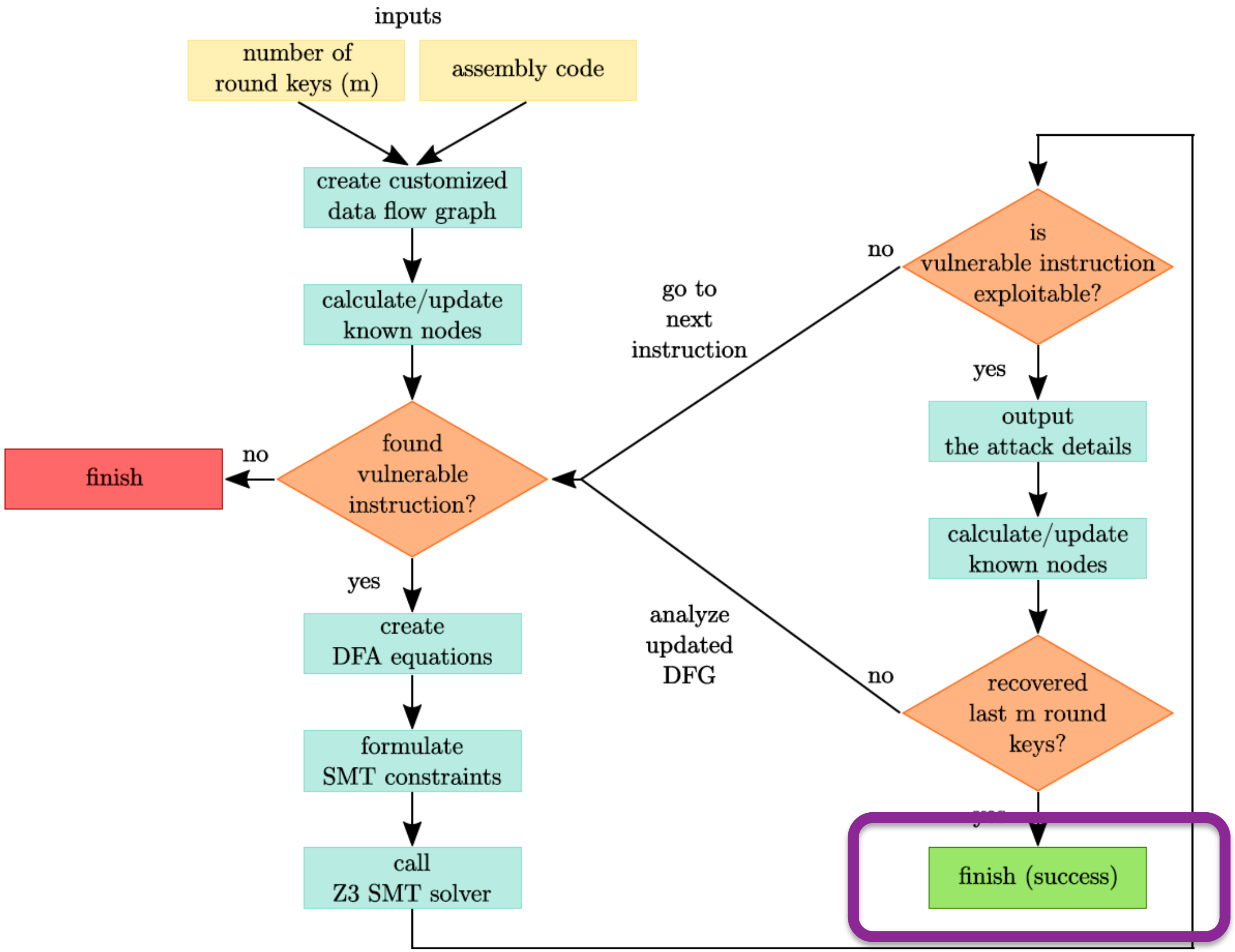


Not yet!

One More Iteration



TADA – Detailed Process Flow



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

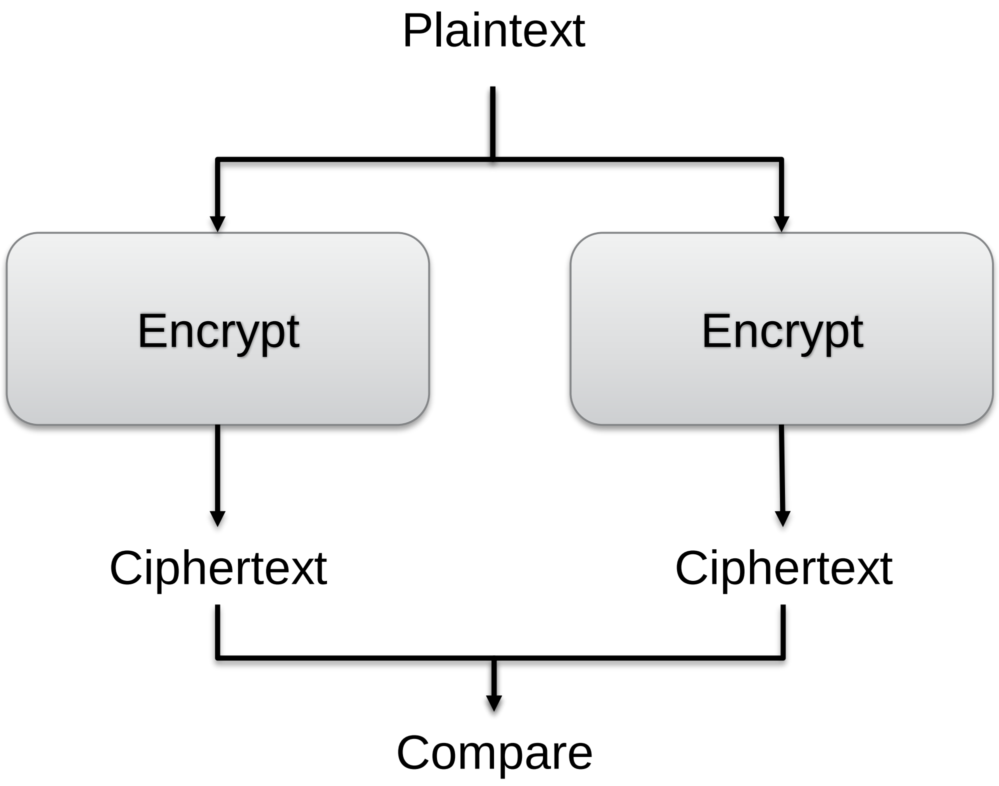
[TBM14] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Differential fault analysis on the families of Simon and Speck ciphers. FDTC 2014.
[Gir05] Christophe Giraud. DFA on AES. Conference on AES 2005.

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



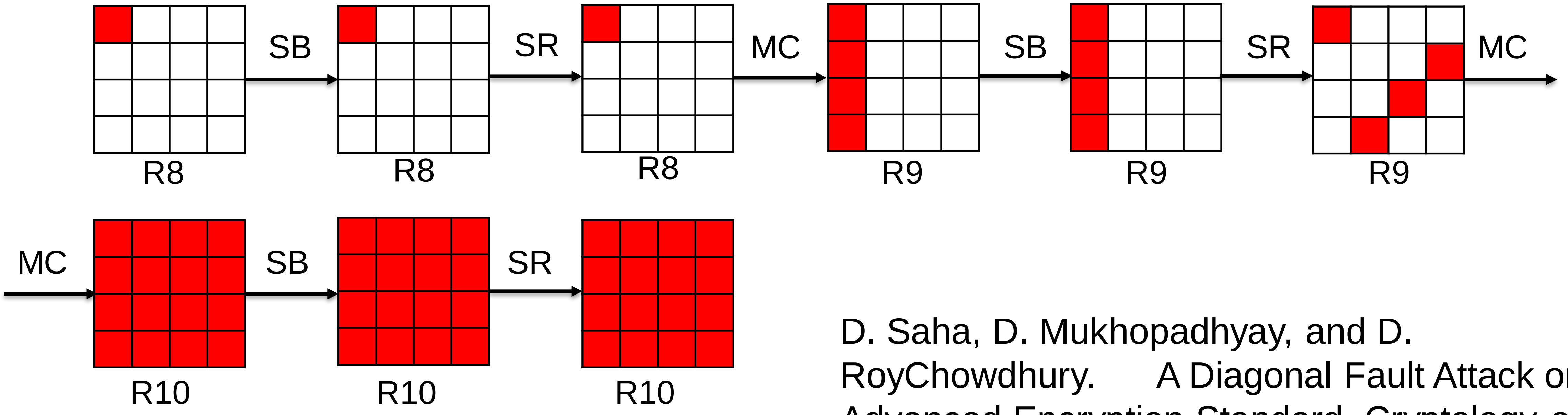
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Countermeasure implementation based on TADA

- After the previous analysis, the *target* and the *vulnerable* nodes change to *target* and *exploitable* nodes – the latter one was proven to be 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

Results – AES

Round	7	8			9			10
# of vulnerable nodes	64	64	48	16	64	48	16	16
Affects # exploitable nodes	4	4	8	16	1	2	4	1



D. Saha, D. Mukhopadhyay, and D. RoyChowdhury. A Diagonal Fault Attack on the Advanced Encryption Standard, Cryptology ePrint Archive: Report 2009/581.

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)

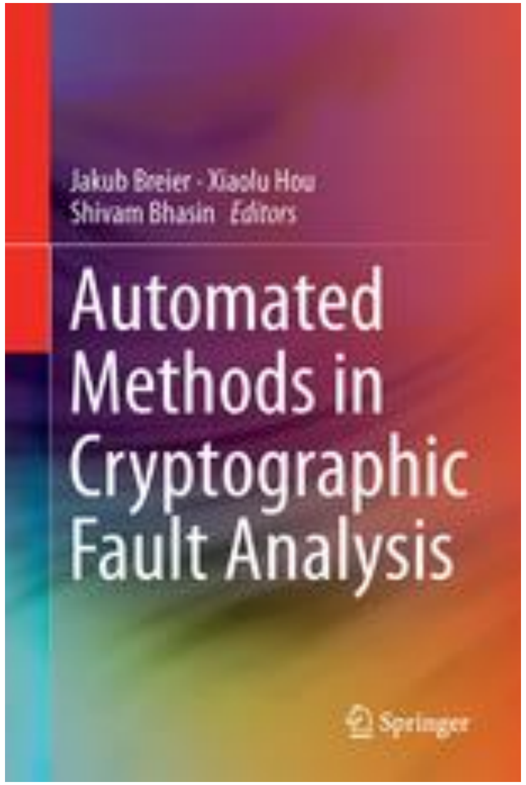
Conclusion

Conclusion

- We showed a way to automate differential fault analysis on block cipher implementations
- Analysis works on a modified data flow graph, vulnerabilities are checked with SMT solver for exploitability
- Countermeasure implementations can be done more efficiently with the support of automated evaluation – number of rounds can be reduced
- For future, it would be good to extend the method to other fault models and other fault analysis techniques

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

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



Thank you for your interest!
Questions?