LOOPLock 3.0: A Robust Cyclic Logic Locking Approach

Pei-Pei Chen, Xiang-Min Yang, Yu-Cheng He, Yung-Chih Chen, Yi-Ting Li, Chun-Yao Wang

Date: 2024/01/24

Presenter: Yu-Cheng He

Outline

- **Introduction**
- Preliminaries
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock 3.0
- Evaluation
- Experimental Results
- Conclusion

Introduction

- In today's global Integrated Circuits (ICs) supply chain, design companies may purchase intellectual property (IP) from IP vendors and integrate them into their designs for saving the development effort.
- To reduce the fabrication cost, they outsource the fabrication to third-party foundries.
- However, the offshore foundries may be untrusted and pose some threats to IP piracy, counterfeiting, and IC overproduction.

Introduction

- **Logic locking** is a useful technique to protect IC designs from potential attackers.
- Its main idea is to use additional key-controlled gates and key inputs to hide the original design.
- The functionality of the locked IC is correct only when the correct key vector is set in the on-chip memory. As a result, attackers cannot pirate the design directly.
- However, most of the traditional logic locking method are vulnerable to the Boolean Satisfiability-based (SAT) Attack.

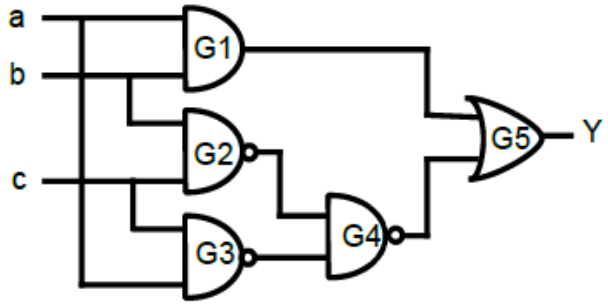


Fig: (a) Original circuit.

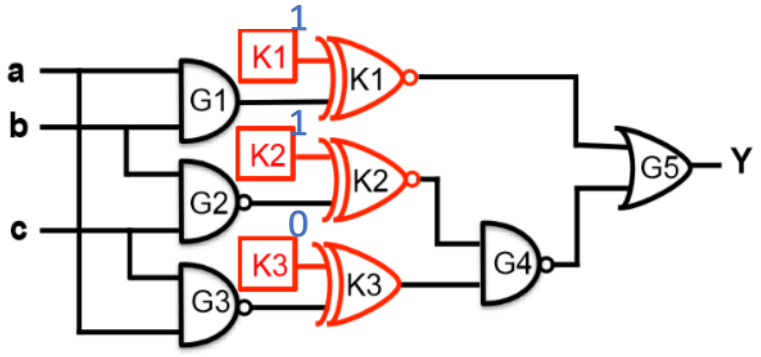


Fig: (b) Logic locking using XOR/XNOR gates.

Outline

- Introduction
- **Preliminaries**
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock 3.0
- Evaluation
- Experimental Results
- Conclusion

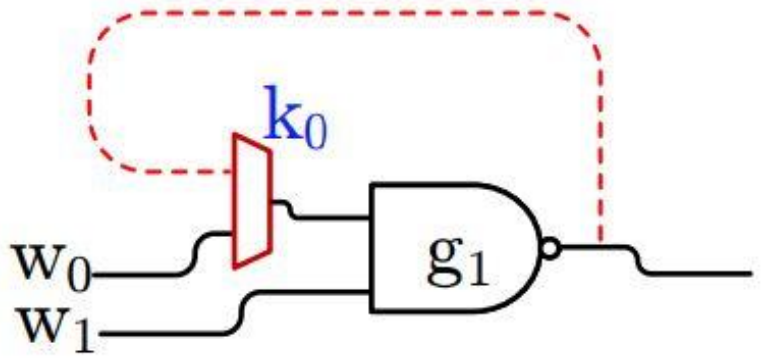
SAT Attack [1]

- SAT attack is an attacking method based on Boolean satisfiability (SAT) algorithms to decrypt several traditional locking methods.
- The components of SAT attack model:
  - A locked circuit
  - A functional correct IC
- SAT attack iteratively rules out incorrect key vectors using **distinguishing input patterns (DIPs)**. A DIP is an input pattern that generates different outputs O_a and O_b under two different key vectors K_a and K_b.
- When no DIP can be found, the remaining key vectors are considered as the correct key vectors.

[1] P. Subramanyan et al., "Evaluating the Security of Logic Encryption Algorithms," in Proc. of HOST, 2015, pp. 137-143.

Cyclic Logic Locking [2]

- Cyclic logic locking is a method that can resist SAT Attack.
- It inserts key gates with feedback edges to cyclify the locked circuit and presents observable non-combinational effects in the primary outputs (POs) under incorrect key vectors.
- The cyclic circuit behaves combinationally only when the correct key vector is fed.



[2] K. Shamsi et al., “Cyclic Obfuscation for Creating SAT-Unresolvable Circuits,” in Proc. of GLSVLSI, 2017, pp. 173-178.

CycSAT [3]

- CycSAT first pre-analyzes the locked netlist to find the **non-cyclic (NC) condition** and then adds the condition to the CNF formula before running the SAT Attack.

- There are two types of CycSAT, CycSAT-I and CycSAT-II, using different constraints to break cycles.
  - CycSAT-I assumes that the original circuit is acyclic, and the NC condition rules out key vectors that make the locked circuit structurally cyclic.

  - CycSAT-II computes the NC condition to break sensitizable cycles and allows the existence of combinational cycles.

[3] H. Zhou et al., “CycSAT: SAT-based attack on cyclic logic encryptions,” in Proc. of ICCAD, 2017, pp. 49-56.

NM-based Cycle Generation [4][5]

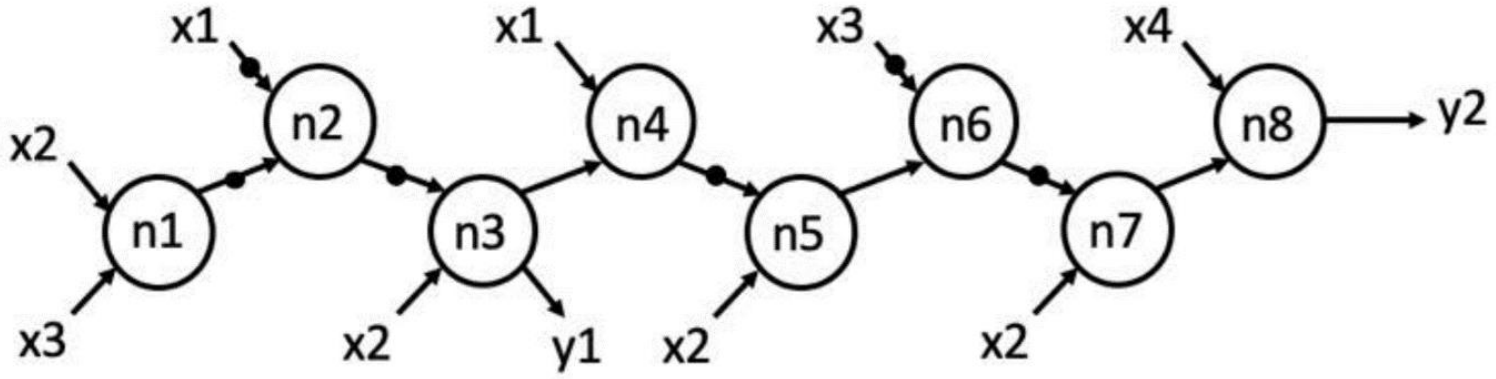
- NM-based cycle generation is a technique to find **cyclic substitute node (CSN)** for a target node, which forms combinational cycles only.
- Let nt denote a target node and ns denote a substitute node in the transitive fanout cone of nt.
- Replacing nt with ns forms a set of cycles C.
- If the value changes on nt are never propagated to ns, C is combinational.
- The functionality of the merged circuit is equivalent to the original one.

[4] Y.-C. Chen and C.-Y. Wang, “Fast Detection of Node Mergers Using Logic Implications,” in Proc. Int. Conf. Computer.-Aided Des., pp. 785-788, 2009.
[5] J.-H. Chen et al., “Synthesis and Verification of Cyclic Combinational Circuits,” in Proc. of SOCC, 2015, pp. 257-262.

NM-based Cycle Generation

- The value change on nt will not propagated to ns under each input pattern.
- A blocking node nb exists between nt and ns, which blocks the effect of the value changes on nt.
- To search for the nb, we can propagate the fault effects from nt to ns, and observe where the fault effects are blocked.

$n4$: blocking node (nb)



$n1$: target node (nt)

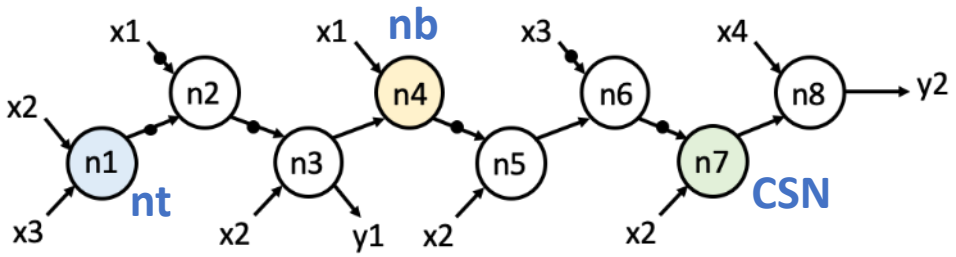
$n7$: substitute node (ns)

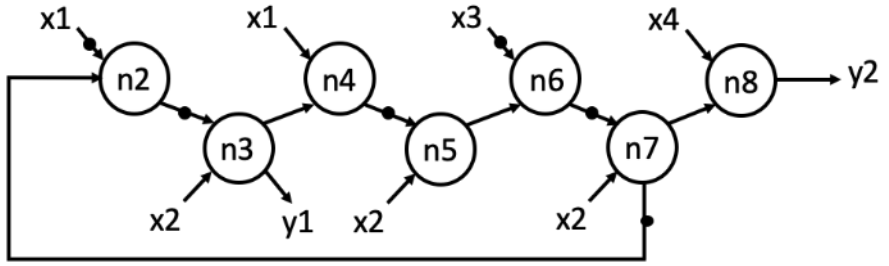
LOOPLock [6]

- LOOPLock is a cyclic logic locking method using NM-based techniques.
- It is able to defend SAT Attack, CycSAT, BeSAT, and Removal Attack.
- Two locking structures, **Type-I cycle pair** and **Type-II cycle pair**, are used to protect the design.
- Each cycle pair deliberately contains **a combinational cycle** and **a non-combinational cycle**.

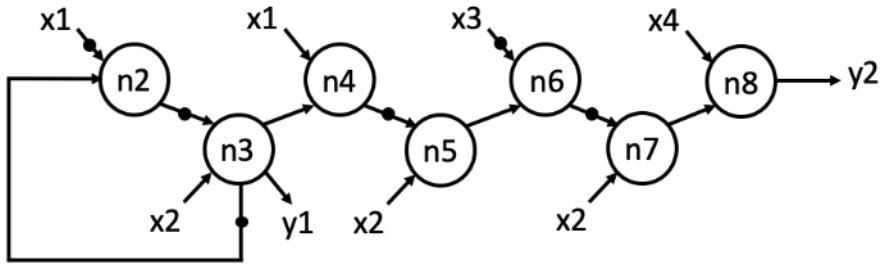
[6] H.-Y. Chiang et al., "LOOPLock: LOgic OPtimization based Cyclic Logic Locking," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2020, vol. 39, no. 10, pp. 2178-2191.

LOOPLock





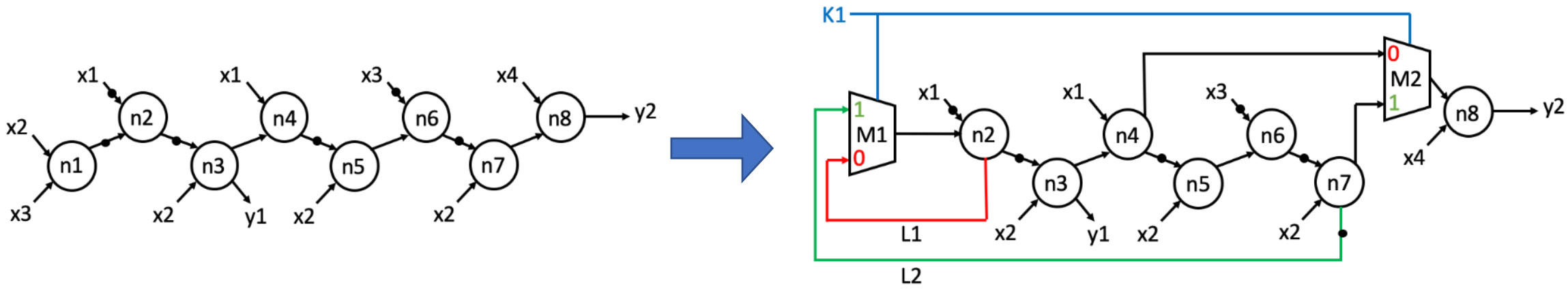
combinational cycle



non-combinational cycle

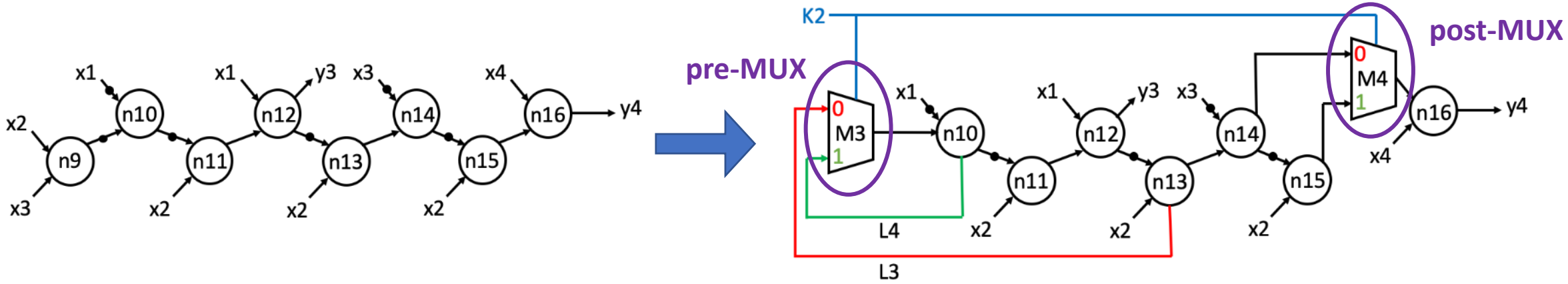
LOOPLock

- Type-I cycle pair
  - The red cycle ($L1$) is a non-combinational cycle affecting POs
  - The green cycle ($L2$) is a functionally *correct* combinational cycle
- Type-I cycle pair is used to defend against SAT attack



LOOPLock

- Type-II cycle pair
  - The red cycle ($L3$) is a combinational cycle, which has no effect on the overall circuit's functionality
  - The green cycle ($L4$) is a non-combinational cycle that is unobservable at POs
- Type-II cycle pair is used to invalidate CycSAT and BeSAT



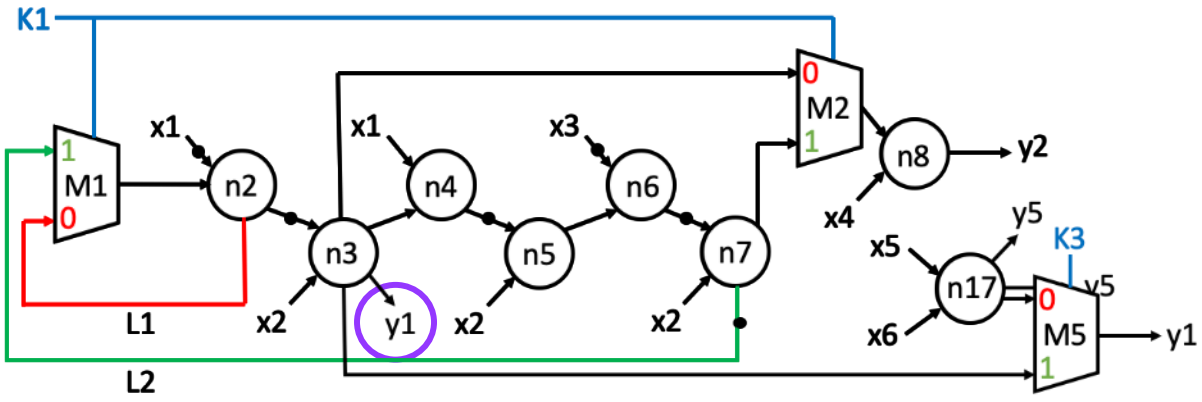
LOOPLock 2.0 [7]

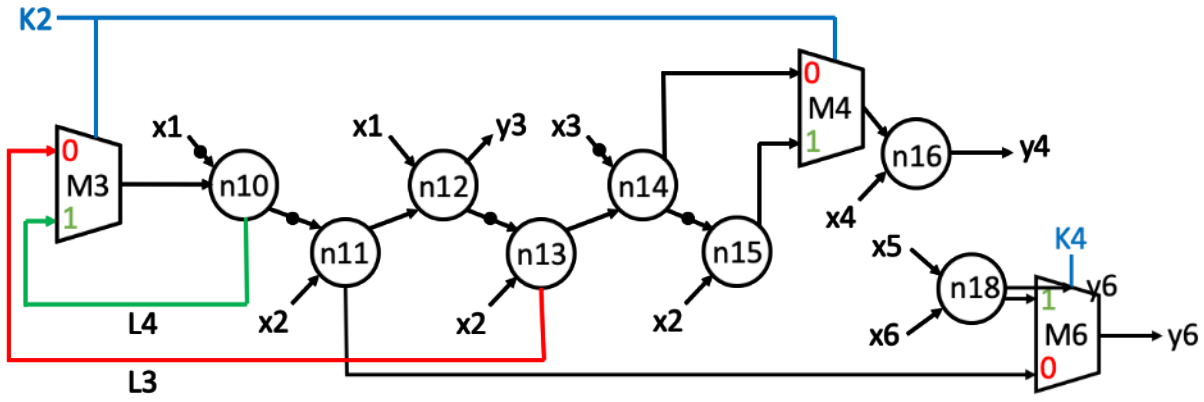
- Unlocking method
  - Blocking Node Identification
    - Remove the MUXs and insert a virtual PI vpi
    - Identify the position of the blocking node nb (**Fault effect propagation**)
  - Identify the type of cycle pair and choose the correct cycle
    - For Type-I cycle pair: There exists some POs between the pre-MUX and nb.
    - For Type-II cycle pair: There is no any PO between the pre-MUX and nb.

[7] X.-M. Yang et al., “LOOPLock 2.0: An Enhanced Cyclic Logic Locking Approach,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2021, doi: 10.1109/TCAD.2021.3053912.

LOOPLock 2.0

- Enhanced structure
  - Let attackers cannot distinguish between Type-I and Type-II cycle pairs by hiding the POs between the pre-MUX and nb.





Outline

- Introduction
- Preliminaries
- **Security Analysis of LOOPLock 2.0**
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock 3.0
- Evaluation
- Experimental Results
- Conclusion

Security Analysis of LOOPLock 2.0

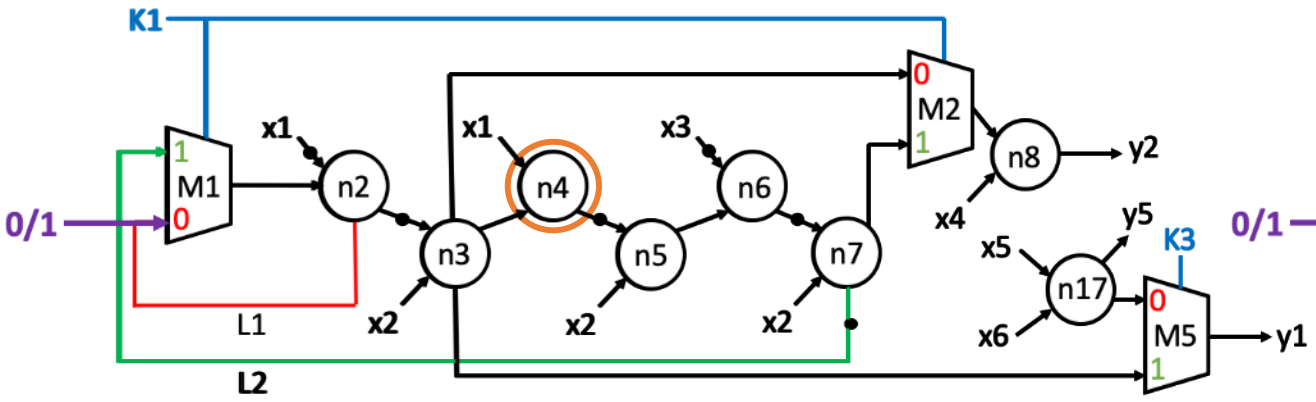
- The shortcomings of LOOPLock 2.0
  - The post-MUX in the Type-I cycle pair lowers the encryption strength.
  - The shared key input in the Type-II cycle pair is used to defend against CycSAT and BeSAT. However, the locking structure may malfunction while the shared key input is split into two individual key inputs.
  - The positions of the blocking nodes nb can still be recognized by removing the key gates and propagating fault effects from the inserted virtual PI, which means that the non-combinational cycles and combinational ones can be distinguished as well.

Outline

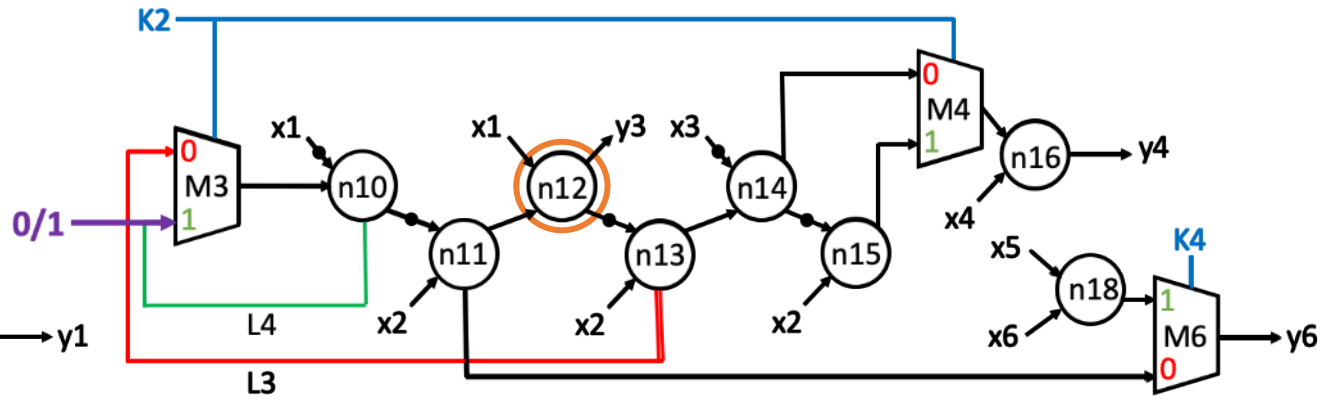
- Introduction
- Preliminaries
- Security Analysis of LOOPLock 2.0
- **An Unlocking Approach to LOOPLock 2.0**
- Our Cyclic Locking Approach: LOOPLock 3.0
- Evaluation
- Experimental Results
- Conclusion

An Unlocking Approach to LOOPLock 2.0 [8]

- According to the last shortcoming of LOOPLock 2.0, the work can distinguish the non-combinational cycles and replacing them with arbitrary constant values (0 or 1).
- Then the correct key can be obtained by applying SAT Attack.



Type-I cycle pair



Type-II cycle pair

[8] P. -P. Chen, X. -M. Yang, Y. -T. Li, Y. -C. Chen and C. -Y. Wang, "An Approach to Unlocking Cyclic Logic Locking - LOOPLock 2.0," 2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2022, pp. 1-7.

Outline

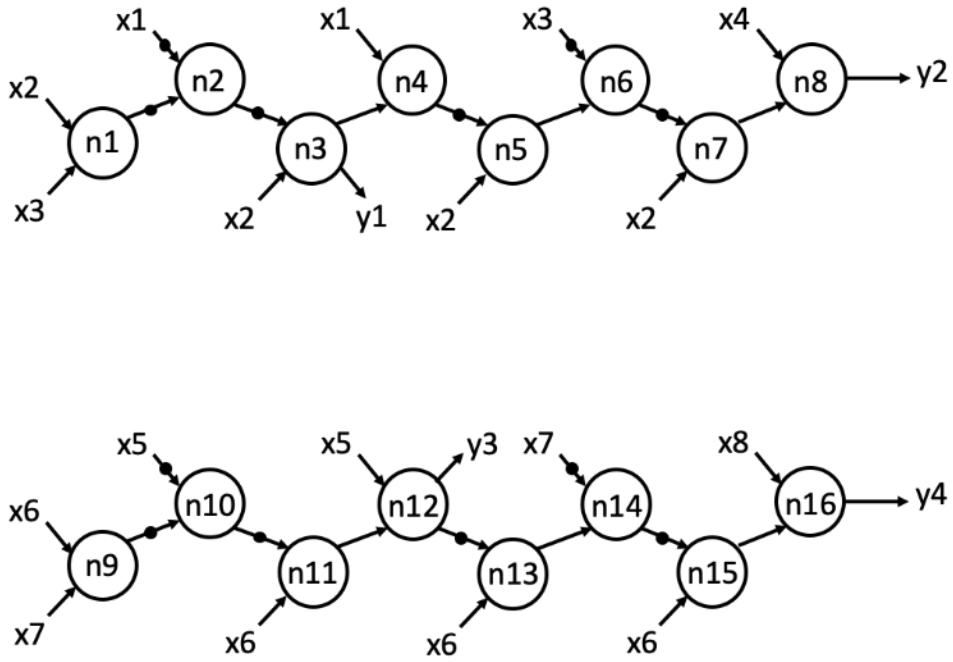
- Introduction
- Preliminaries
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- **Our Cyclic Locking Approach: LOOPLock 3.0**
- Evaluation
- Experimental Results
- Conclusion

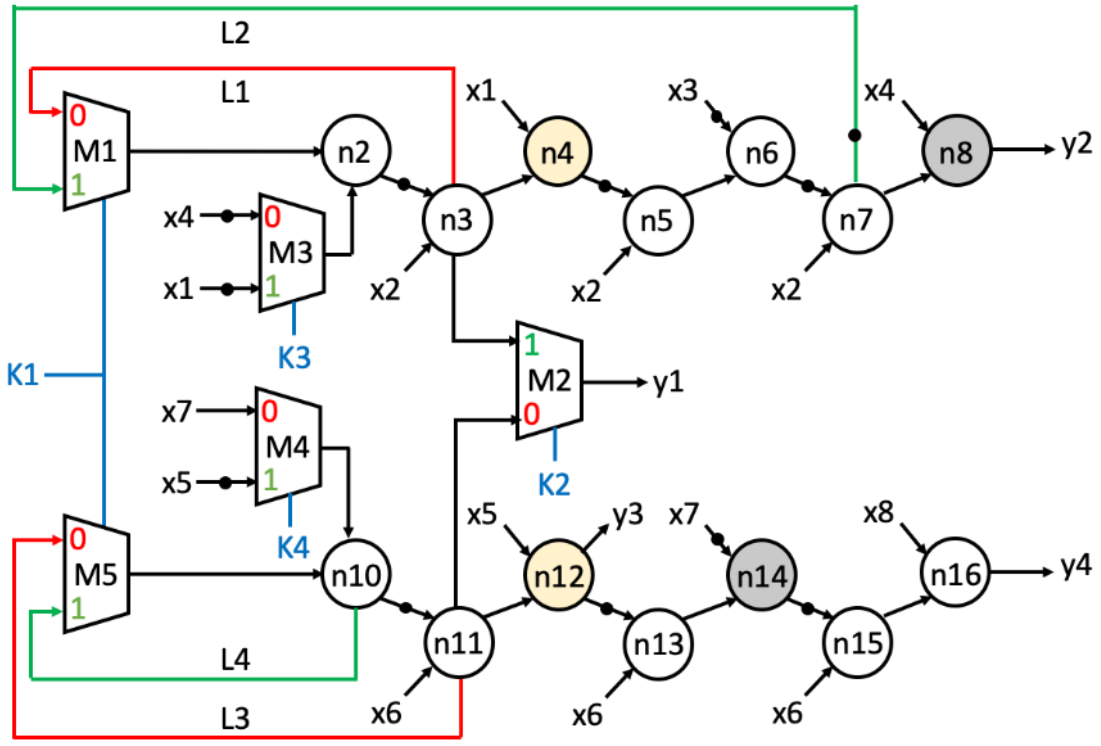
LOOPLock 3.0

- We present a new cyclic locking structure considering the shortcomings of LOOPLock 2.0
  - The post-MUX in the Type-I cycle pair
  - The shared key input controls the pre-MUX and the post-MUX
  - The position of nb

- The enhanced structure is to hide the position of nb and make attackers unable to get correct key values even splitting the shared key input.

LOOPLock 3.0

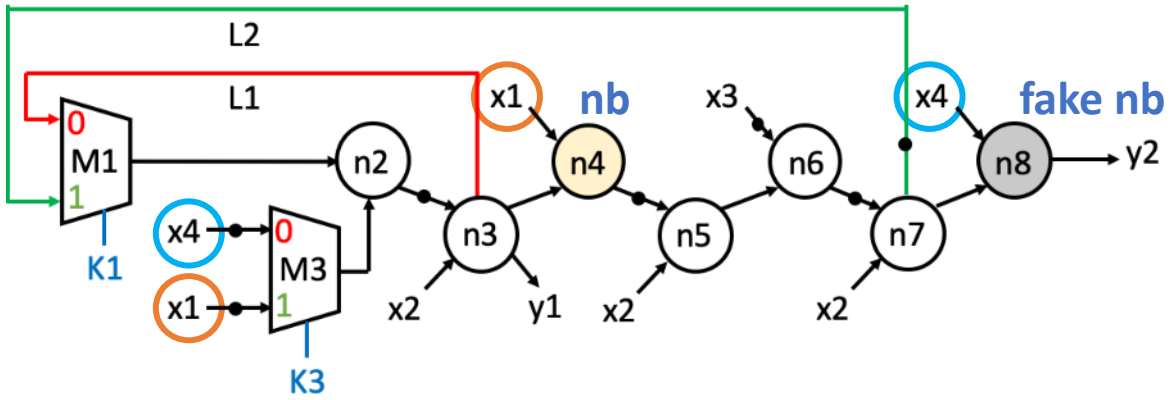




The correct key: $(K1, K2, K3, K4) = (1,1,1,1)$

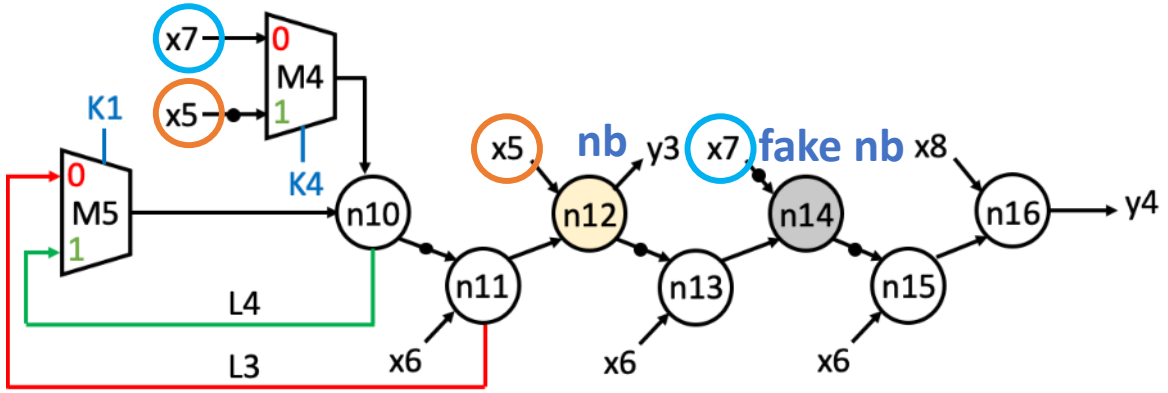
LOOPLock 3.0

- Type-I cycle pair
  - The red cycle ($L1$) is a non-combinational cycle affecting POs
  - The green cycle ($L2$) is a functionally *correct* combinational cycle
- MUX $M3$ is used to hide the position of nb ($n4$)
  - $M3$ is inserted at $n2$ prior to nb whose side input is opposite to nb's side input
  - $n8$ is a fake nb, which is used to obfuscate attackers



LOOPLock 3.0

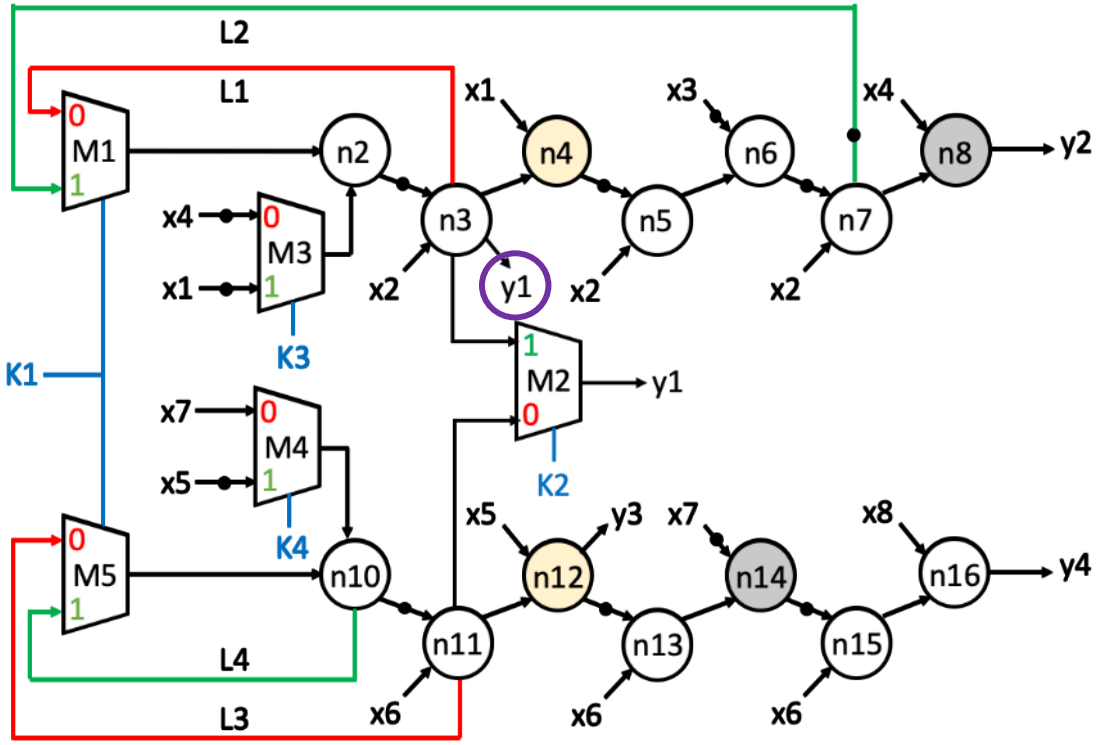
- Type-III cycle pair
  - Both cycles ($L3$, $L4$) are non-combinational cycles, which are unobservable at POs
- MUX $M4$ is used to hide the position of nb ($n12$)
- $NC_{L3} = K1$ and $NC_{L4} = \overline{K1}$, so the NC condition of the Type-III cycle pair is $K1 \wedge \overline{K1}$, which will lead to contradiction while deriving the NC condition of CycSAT and BeSAT.



LOOPLock 3.0

- $K1$ controls the key gates of two cycle pairs ($M1$, $M5$) to select two cycles simultaneously.

- $K2$ is used for hiding the structural difference between the Type-I and Type-III cycle pairs and propagating the non-combinational effect in the Type-III cycle pair to the PO $y1$.

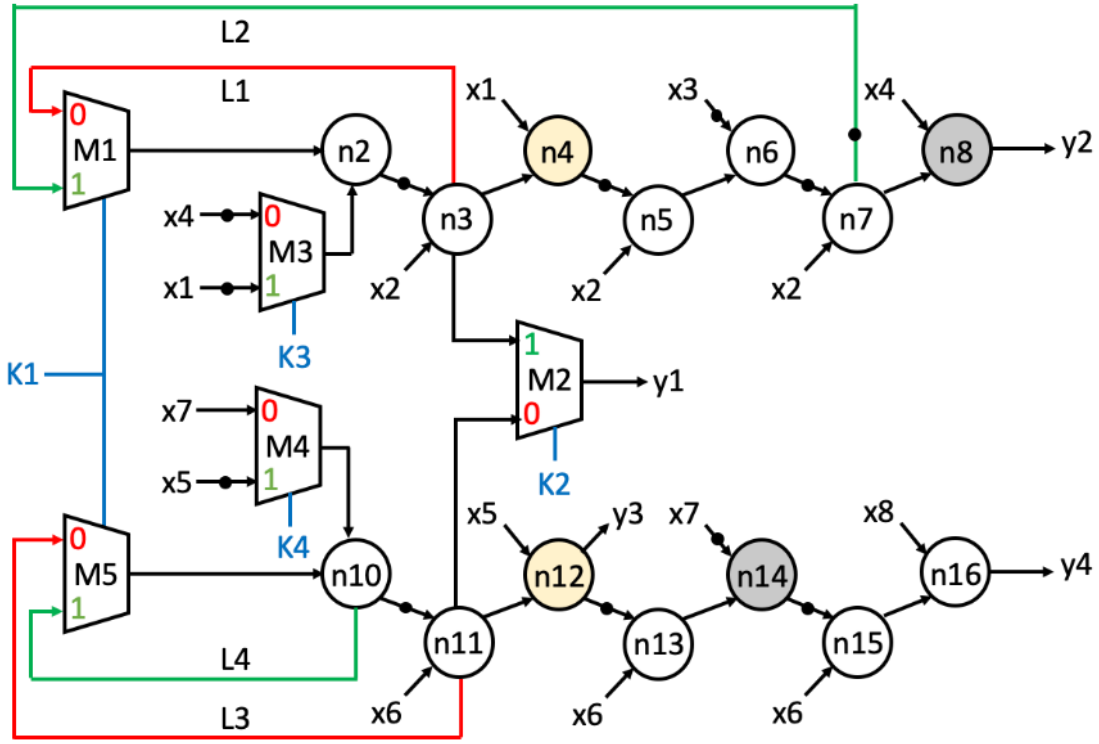


Outline

- Introduction
- Preliminaries
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock 3.0
- **Evaluation**
- Experimental Results
- Conclusion

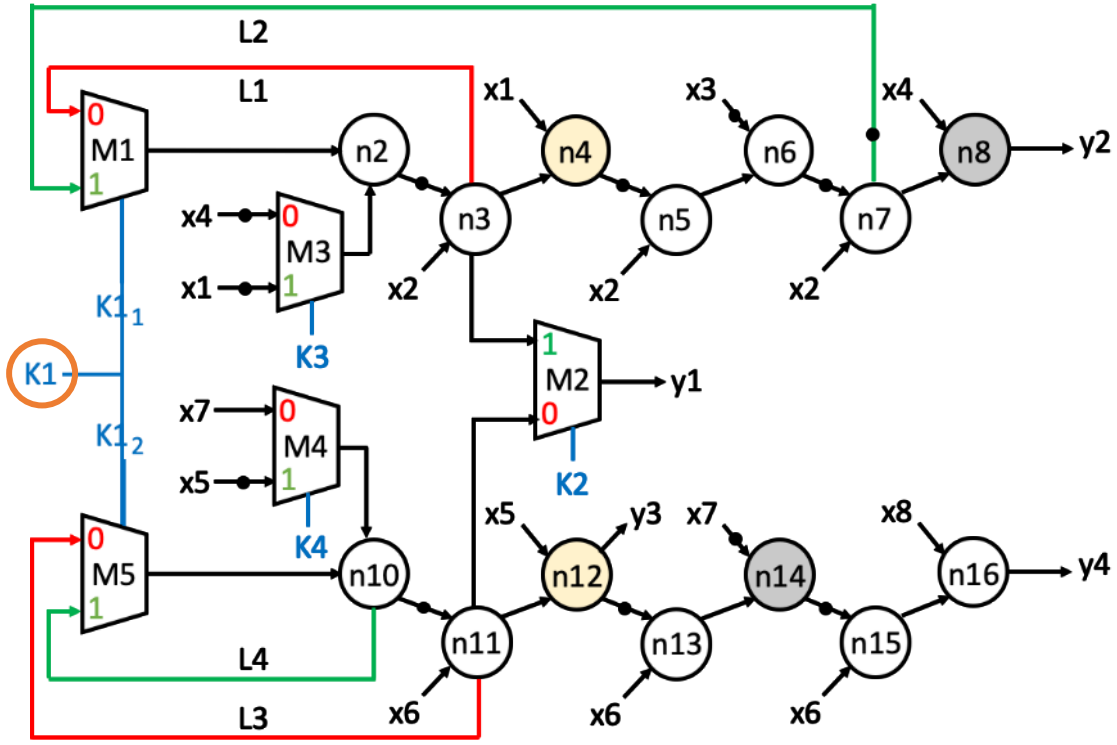
Evaluation

- For the first shortcoming, the post-MUX in the Type-I cycle pair in LOOPLock 2.0 may lower the encryption strength. Since we only keep the pre-MUX of the Type-I cycle pair to invalidate SAT Attack, this shortcoming disappears in the proposed locking structure.



Evaluation

- For the second shortcoming, LOOPLock 2.0 can be unlocked by the method of key-splitting. Although we use the shared key input $K1$ in our locking structure, it is still effective against CycSAT and BeSAT while $K1$ is split into two key inputs $K1_1$ and $K1_2$.



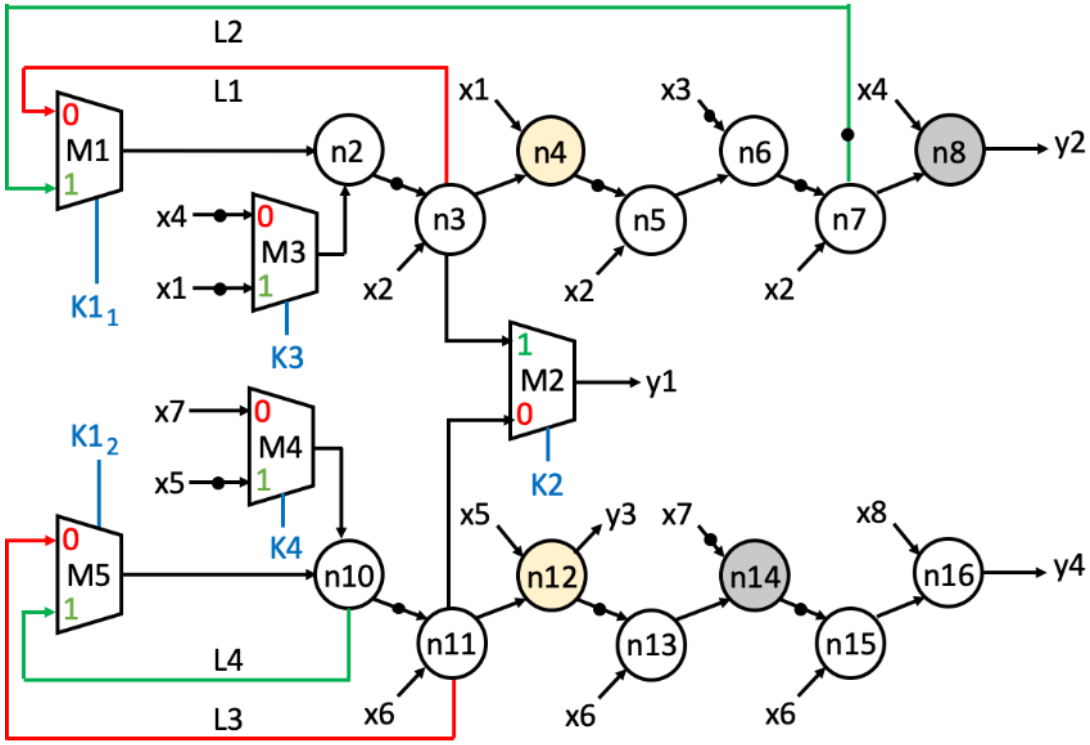
Evaluation

- **Applying CycSAT-II on the proposed locking approach with split key inputs**

- $NC_{L1} = K1_1 \vee (K3 \wedge x1) \vee (\overline{K3} \wedge x4) \vee \overline{x2}$
- $NC_{L2} = \overline{K1_1} \vee (K3 \wedge x1) \vee (\overline{K3} \wedge x4) \vee \overline{x2} \vee \overline{x1} \vee x3$
- $NC_{L3} = K1_2 \vee (K4 \wedge x5) \vee (\overline{K4} \wedge \overline{x7}) \vee \overline{x6}$
- $NC_{L4} = \overline{K1_2} \vee (K4 \wedge x5) \vee (\overline{K4} \wedge \overline{x7})$

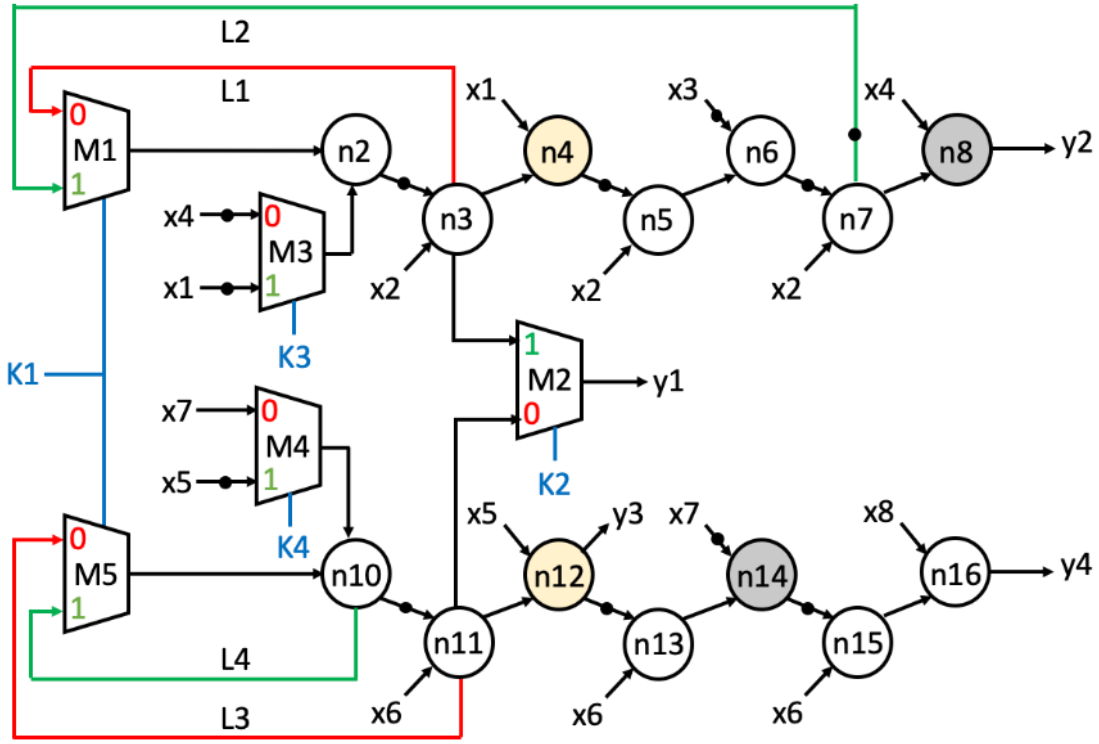
- $NC_{L1}(x1 = 0, x2 = 1, x4 = 0) = K1_1$
- $NC_{L2}(x1 = 1, x2 = 1, x3 = 0, x4 = 0) = \overline{K1_1} \vee K3$
- $NC_{L3}(x5 = 0, x6 = 1, x7 = 1) = K1_2$
- $NC_{L4}(x5 = 0, x7 = 1) = \overline{K1_2}$

- $NC\ condition = K1_1 \wedge (\overline{K1_2} \vee K3) \wedge \boxed{K1_2 \wedge \overline{K1_2}} = 0$



Evaluation

- For the last shortcoming, the position of nb is identifiable in LOOPLock 2.0. However, in the proposed locking structure, we use key gates $M3$ and $M4$ to hide the position of the real nb.



Outline

- Introduction
- Preliminaries
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock3.0
- Evaluation
- **Experimental Results**
- Conclusion

Experimental Results

- LOOPLock 3.0 was implemented in C language
- The experiments about the locking approach were conducted on an Intel Xeon E5-2650v2 2.60GHz CentOS 6.10 platform with 64GBytes memory

- Benchmarks were from http://iwls.org/iwls2005/benchmarks.html

TABLE I: Results of the proposed locking approach in identifying all the locking structures.

Benchmark	|PI|/|PO|	|Node|	|Type-I|	|Type-III|	|Lock|
aes_core	789/659	21513	1844	228	228
b17	1454/1512	52920	126	392	126
b20	522/512	12219	66	129	66
b21	522/512	12782	60	135	60
b22	767/757	18488	98	197	98
C1908	33/25	414	8	24	8
C3540	50/22	1038	57	23	23
C432	36/7	206	12	16	12
C5315	178/123	1773	6	6	6
C7552	207/107	2074	18	27	18
dalu	75/16	1740	10	38	10
des_area	368/192	4857	2	2	2
i10	257/224	2673	146	59	59
i2c	147/142	1306	2	8	2
i8	133/81	3310	67	73	67
mem_ctrl	1198/1235	15641	176	173	173
pci_brdge32	3521/3566	24369	45	142	45
pci_spoci_ctrl	85/73	1451	14	32	14
rot	135/107	1063	9	23	9
s13207	700/790	2719	11	62	11
s38417	1664/1742	9219	86	204	86
s38584	1464/1730	12400	31	133	31
s9234	247/250	1958	14	43	14
sasc	133/129	784	3	8	3
systemcaes	930/799	13054	35	138	35
tv80	373/391	9609	413	220	220
usb_funct	1874/1867	15894	23	147	23
wb_conmax	1900/2186	48429	339	376	339
Avg.	-	-	132.89	109.21	63.86

TABLE II: Results of the proposed locking approach about Area-Delay-Product (ADP).

Benchmark	|Lock|	|Original Node|	|Locked Node|	|Original Level|	|Locked Level|	ADP
aes_core	5	21513	21642 (1.01)	26	70 (2.69)	2.71
b17	5	52920	53027 (1.00)	43	52 (1.21)	1.21
b20	5	12219	12323 (1.01)	66	91 (1.38)	1.39
b21	5	12782	12887 (1.01)	67	77 (1.15)	1.16
b22	5	18488	18598 (1.01)	69	87 (1.26)	1.27
C1908	5	414	513 (1.24)	32	92 (2.88)	3.56
C3540	5	1038	1132 (1.09)	41	65 (1.59)	1.73
C432	5	206	309 (1.50)	42	84 (2.00)	3.00
C5315	5	1773	1885 (1.06)	38	89 (2.34)	2.49
C7552	5	2074	2173 (1.05)	29	69 (2.38)	2.49
dalu	5	1740	1820 (1.05)	39	44 (1.13)	1.18
des_area	2	4857	4895 (1.01)	33	45 (1.36)	1.37
i10	5	2673	2775 (1.04)	51	64 (1.25)	1.30
i2c	2	1306	1334 (1.02)	16	20 (1.25)	1.28
i8	5	3310	3410 (1.03)	27	27 (1.00)	1.03
mem_ctrl	5	15641	15729 (1.01)	36	44 (1.22)	1.23
pci_brdge32	5	24369	24449 (1.00)	31	32 (1.03)	1.04
pci_spoci_ctrl	5	1451	1548 (1.07)	19	63 (3.32)	3.54
rot	5	1063	1165 (1.10)	51	61 (1.20)	1.31
s13207	5	2719	2824 (1.04)	34	41 (1.21)	1.25
s38417	5	9219	9293 (1.01)	30	37 (1.23)	1.24
s38584	5	12400	12498 (1.01)	36	50 (1.39)	1.40
s9234	5	1958	2063 (1.05)	36	59 (1.64)	1.73
sasc	3	784	839 (1.07)	9	27 (3.00)	3.21
systemcaes	5	13054	13150 (1.01)	47	55 (1.17)	1.18
tv80	5	9609	9708 (1.01)	52	52 (1.00)	1.01
usb_funct	5	15894	15894 (1.00)	27	50 (1.85)	1.85
wb_conmax	5	48429	48536 (1.00)	27	74 (2.74)	2.75

TABLE III: The result of applying attacking methods on the circuits locked by the proposed locking approach.

Benchmark Information		[8]		SAT Attack		CycSAT		BeSAT	
Benchmark	|Lock|	Time (s)	Result	Time (s)	Result	Time (s)	Result	Time (s)	Result
aes_core	5	Inf. loop	No Result	Inf. loop	No Result	1.596	No Result	1.414	No Result
b17	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	46.414	No Result
b20	5	Inf. loop	No Result	Inf. loop	No Result	0.836	No Result	0.903	No Result
b21	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	1355.084	Wrong Key
b22	5	Inf. loop	No Result	Inf. loop	No Result	1.128	No Result	1.160	No Result
C1908	5	0.147	No Result	0.399	No Result	0.178	No Result	0.164	No Result
C3540	5	0.697	Wrong Key	Inf. loop	No Result	Inf. loop	No Result	2.263	No Result
C432	5	0.070	No Result	Inf. loop	No Result	0.077	No Result	0.078	No Result
C5315	5	0.585	No Result	0.195	No Result	1.021	No Result	0.479	No Result
C7552	5	0.156	No Result	0.157	No Result	0.379	No Result	0.168	No Result
dalu	5	13.434	Wrong Key	Inf. loop	No Result	0.131	No Result	0.124	No Result
des_area	2	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	6.021	Wrong Key
i10	5	0.296	No Result	Inf. loop	No Result	0.379	No Result	0.384	No Result
i2c	2	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	0.183	No Result
i8	5	Inf. loop	No Result	0.297	No Result	0.324	No Result	0.305	No Result
mem_ctrl	5	Inf. loop	No Result	Inf. loop	No Result	15.495	No Result	3.054	No Result
pci_brdge32	5	47.821	No Result	Inf. loop	No Result	Inf. loop	No Result	6697.215	out of memory
pci_spoci_ctrl	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	timeout	No Result
rot	5	0.078	No Result	0.171	No Result	0.055	No Result	0.057	No Result
s13207	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	593.289	No Result
s38417	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	110.990	No Result
s38584	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	42.375	No Result
s9234	5	0.313	Wrong Key	Inf. loop	No Result	0.492	No Result	0.496	No Result
sasc	3	0.091	No Result	Inf. loop	No Result	0.058	No Result	0.052	No Result
systemcaes	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	62.904	No Result
tv80	5	Inf. loop	No Result	0.659	No Result	0.663	No Result	0.632	No Result
usb_funct	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	47.230	No Result
wb_conmax	5	Inf. loop	No Result	Inf. loop	No Result	Inf. loop	No Result	468.508	No Result

Outline

- Introduction
- Preliminaries
- Security Analysis of LOOPLock 2.0
- An Unlocking Approach to LOOPLock 2.0
- Our Cyclic Locking Approach: LOOPLock3.0
- Evaluation
- Experimental Results
- **Conclusion**

Conclusion

- A new cyclic logic locking structure is presented in this work to enhance the security level of the locked circuit.

- The experimental results show that the proposed locking approach can effectively defend against SAT Attack, CycSAT, BeSAT, and the previous proposed unlocking approach.