SAT Attack Complexity Analysis

Yadi Zhong and **Ujjwal Guin**

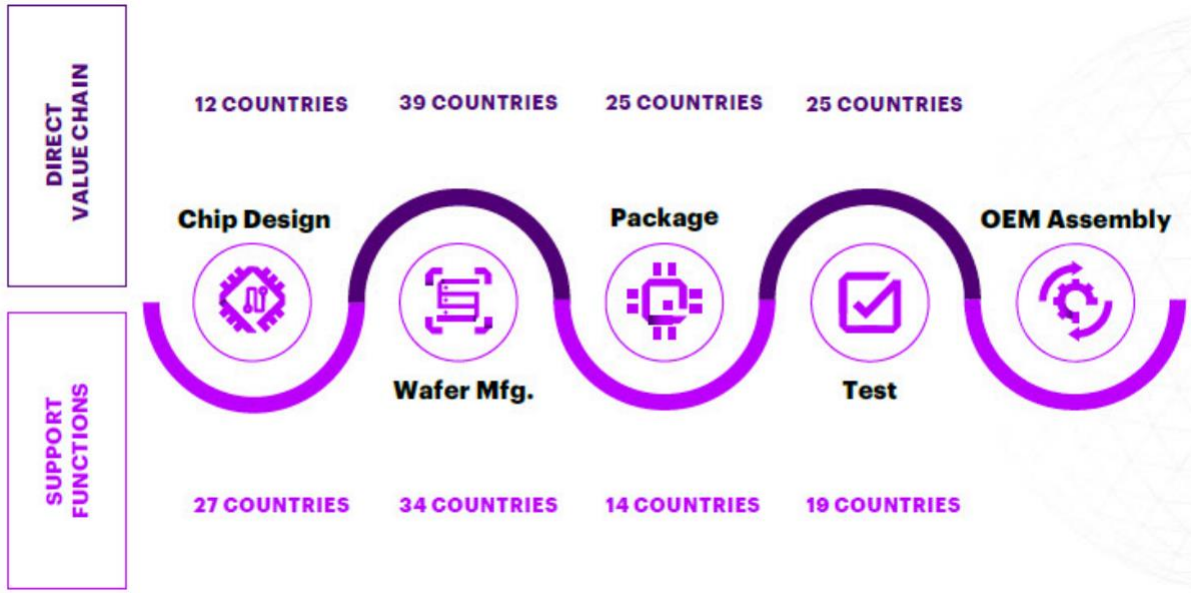
Auburn University, AL, USA

July 9, 2022

CAD4Sec Workshop

Motivation

- Globalized semiconductor manufacturing and test[1-2]
  - Diminishing share of U.S. semiconductor manufacturing
    - Increasing reliance on offshore foundries.
  - Malicious supply chain disruptions.
  - Rise of IP theft.
  - $412 billion semiconductor industry is at risk.
  - Logic Locking is the future.



Number of countries with enterprises participating in various phases of semiconductor production activity[2].

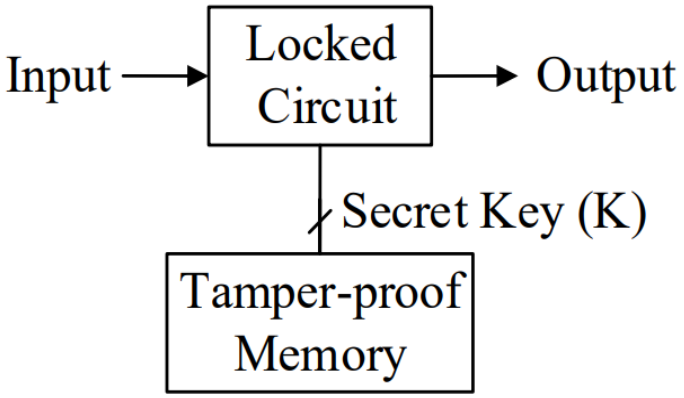
[1] “White House 100-Day Reviews under Executive Order 14017" on "Building Resilient Supply Chains, Revitalizing American Manufacturing, and Fostering Broad-Based Growth," June 2021.
[2] Moore’s Law Under Attack: The Impact of China’s Policies on Global Semiconductor Innovation, 2021

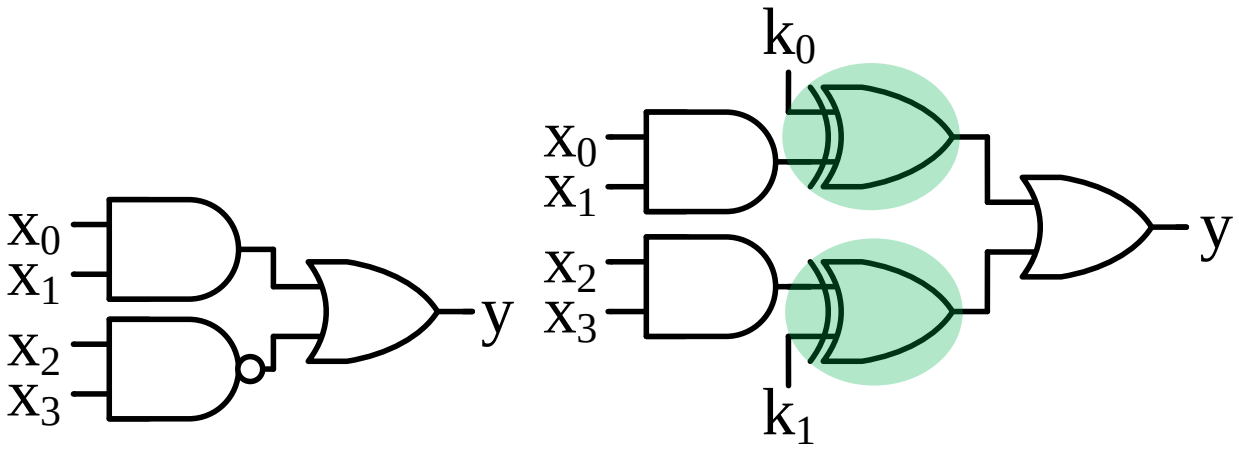
Outline

- Overview – Logic Locking
- Background – Boolean Satisfiability
- SAT Attack Complexity Analysis
  - Traditional XOR-based Locking
  - Point Function-based Locking
- Time Complexity Analysis for Traditional SAT Attack
- Conclusion

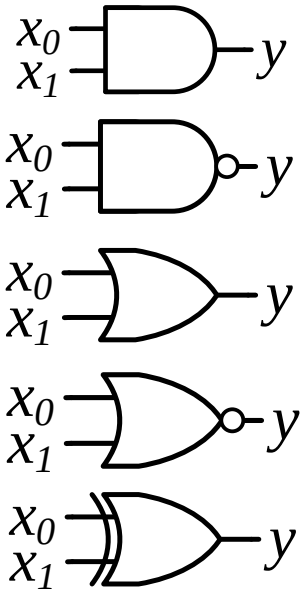
Overview of Logic Locking

- Obfuscate the inner details of a circuit.
- The correct functionality is:
  - preserved when a correct key is programmed in the tamper-proof memory.
  - altered for some input patterns when a wrong key is applied.
- It is impossible to determine the key bit just simply looking at a key gate.





Boolean Satisfiability – Conjunctive Normal Form (CNF)



AND	$y = x_0 \cdot x_1$	$(\overline{x_0} \vee \overline{x_1} \vee y) \wedge (x_0 \vee \bar{y}) \wedge (x_1 \vee \bar{y})$
NAND	$y = \overline{x_0 \cdot x_1}$	$(\overline{x_0} \vee \overline{x_1} \vee \bar{y}) \wedge (x_0 \vee y) \wedge (x_1 \vee y)$
OR	$y = x_0 + x_1$	$(x_0 \vee x_1 \vee \bar{y}) \wedge (\overline{x_0} \vee y) \wedge (\overline{x_1} \vee y)$
NOR	$y = \overline{x_0 + x_1}$	$(x_0 \vee x_1 \vee y) \wedge (\overline{x_0} \vee y) \wedge (\overline{x_1} \vee y)$
XOR	$y = x_0 \oplus x_1$	$(\overline{x_0} \vee \overline{x_1} \vee \bar{y}) \wedge (x_0 \vee x_1 \vee \bar{y})$ $\wedge (x_0 \vee \overline{x_1} \vee y) \wedge (\overline{x_0} \vee x_1 \vee y)$

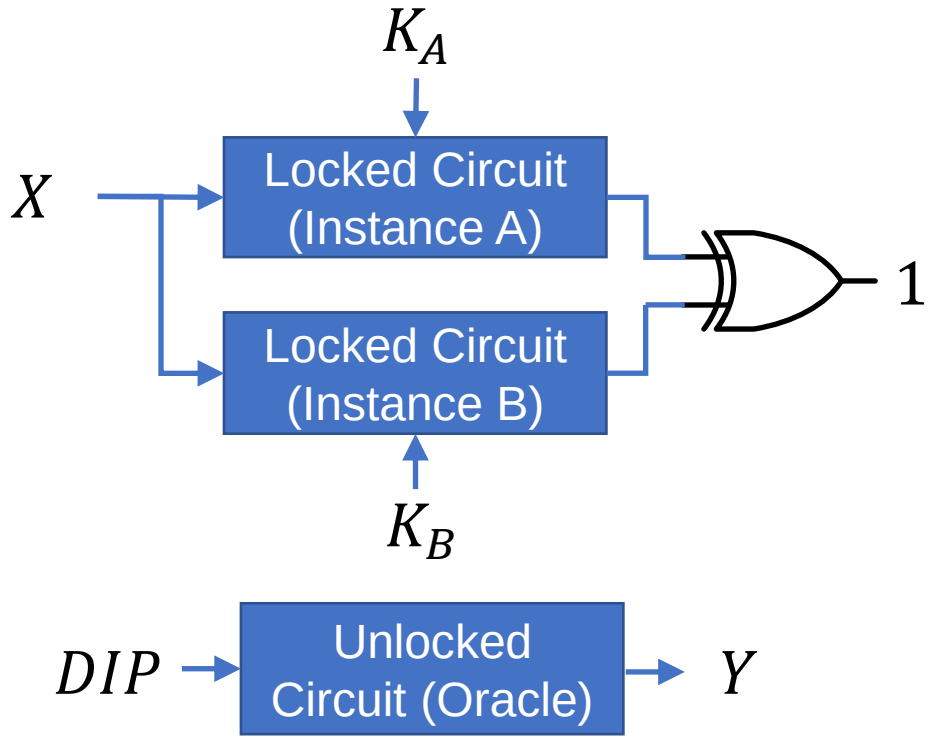
Terminology:

- Literals: x_i, y
- Clauses: $(\overline{x_0} \vee \overline{x_1} \vee y)$, $(x_0 \vee \bar{y})$, …
- Conjunctive normal form (CNF): $(\overline{x_0} \vee \overline{x_1} \vee y) \wedge (x_0 \vee \bar{y}) \wedge (x_1 \vee \bar{y})$

Background – SAT Attack

- Finding the distinguishing input pattern (DIP) from the miter circuit.
- Deriving the correct key: CNF update.
- Reporting DIPs and key value.
  - UNSAT at the last iteration.

```
3  while (true) do
4  |   i ← i + 1 ;
5  |   [X_i, K_i, r] = sat[F ∧ (Y_{A_i} ≠ Y_{B_i})];
6  |   if (r == false) then
7  |   |   break;
8  |   end
9  |   Y_i = sim_eval(X_i);
10 |   F ← F ∧ C(X_i, K, Y_i);
11 end
```



Background – Post-SAT Solutions

- Post-SAT locking techniques:
  - Target exponential iterations
  - Time out for SAT attack
- Novel attacks emerge to break these logic locking.

Table: Summary of Post-SAT locking and Attacks

Locking Type	Techniques	Attacks
Point function	[23]–[32]	[71]–[85]
Cyclic	[33]–[38]	[86]–[88]
LUT	[39]–[46]	[46], [89], [90]
Scan	[47]–[52]	[91]–[93]
FSM	[53]–[58]	[94]–[99]
Timing	[59]–[64]	[100], [101]
HLS	[65]–[70]	[83], [84], [101]

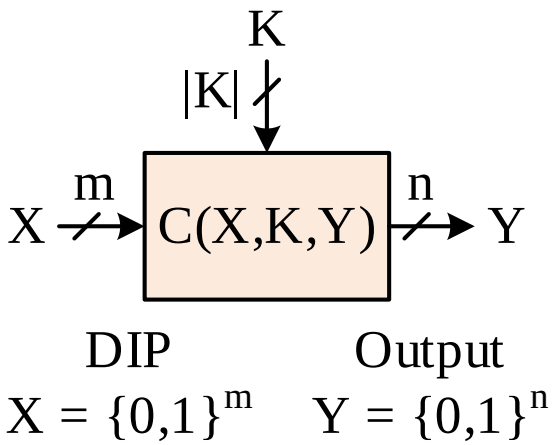
[1]. Y. Zhong and U. Guin, “Complexity Analysis of the SAT Attack on Logic Locking,” arXiv preprint arXiv:2207.01808, 2022.

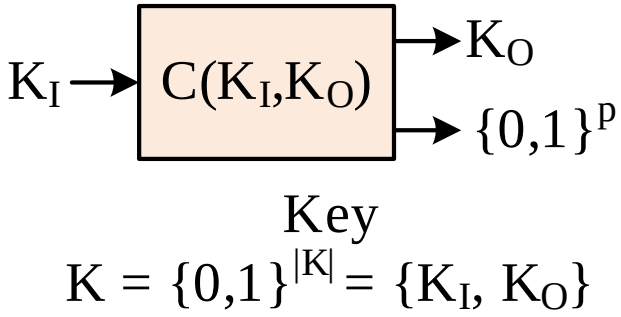
Outline

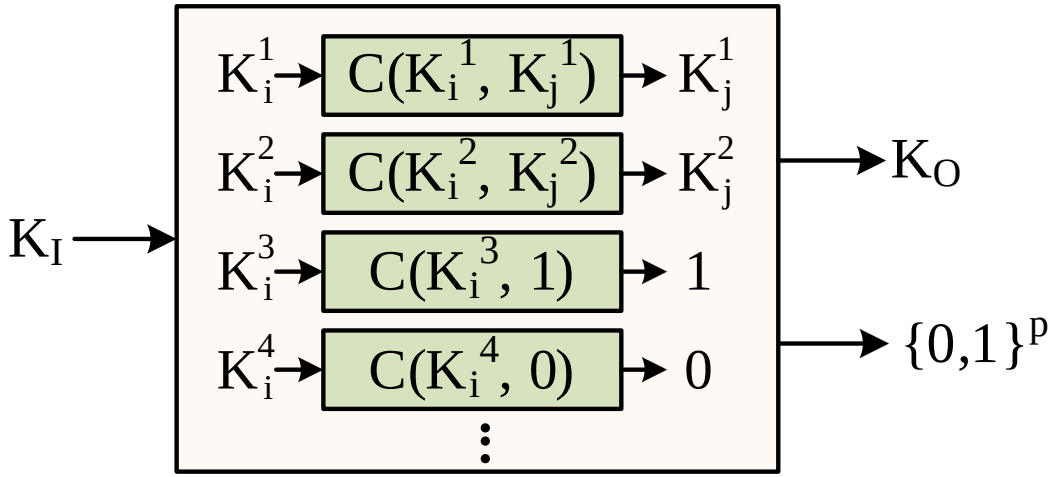
- Overview
- Background
- SAT Attack Complexity Analysis
  - Traditional XOR-based Locking
  - Point Function-based Locking
- Time Complexity Analysis for Traditional SAT Attack
- Conclusion

SAT Attack Complexity Analysis: Traditional XOR-based Locking

- The DIP X and its corresponding oracle output Y forms an IO pair.
- An IO pair $\{X, Y\}$ reduces the locked circuit $C(X, K, Y)$ to functions of keys $C(K_I, K_O)$

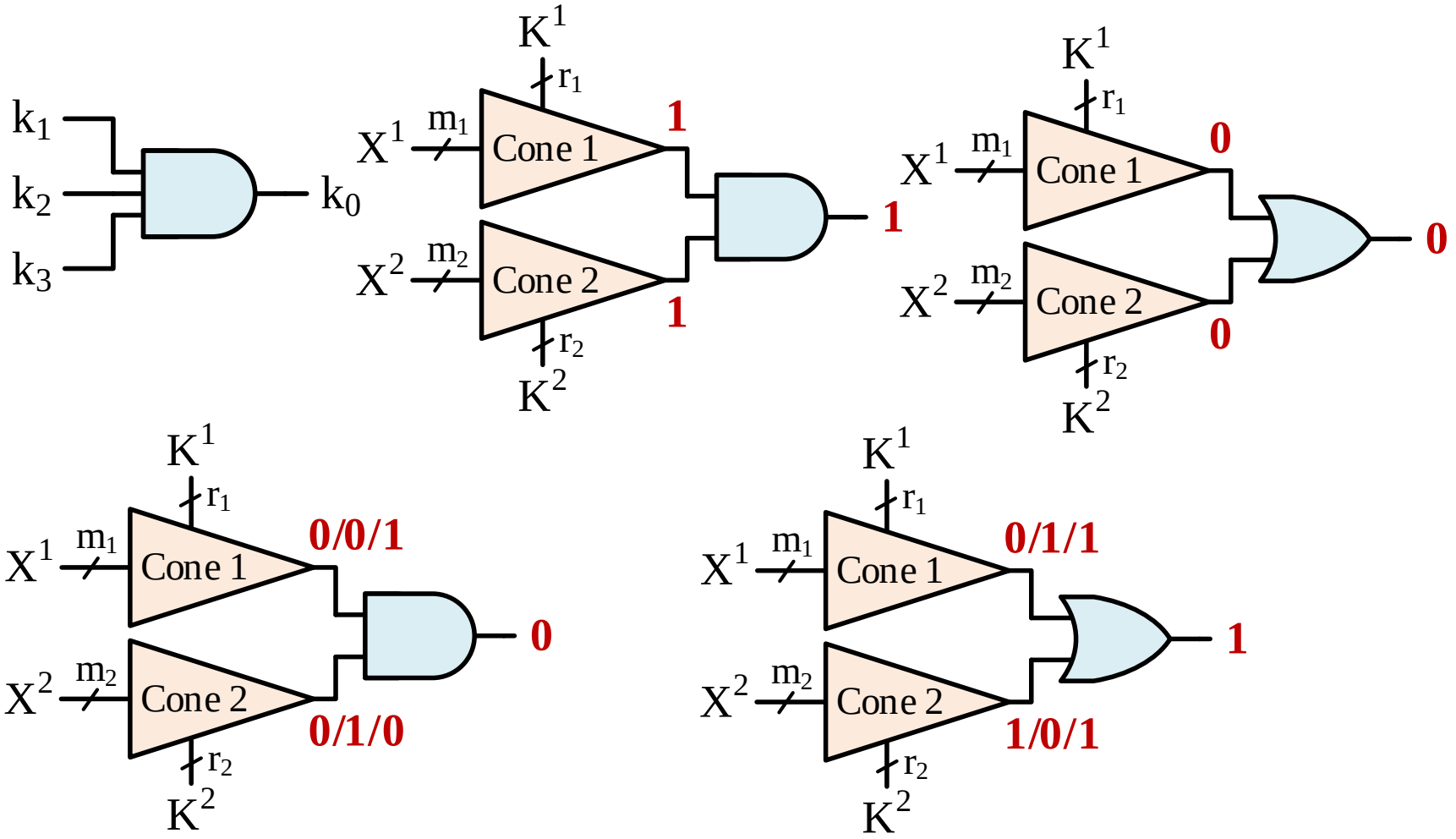




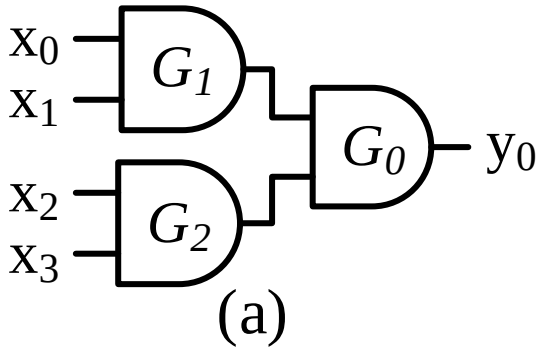


Key Pruning Analysis

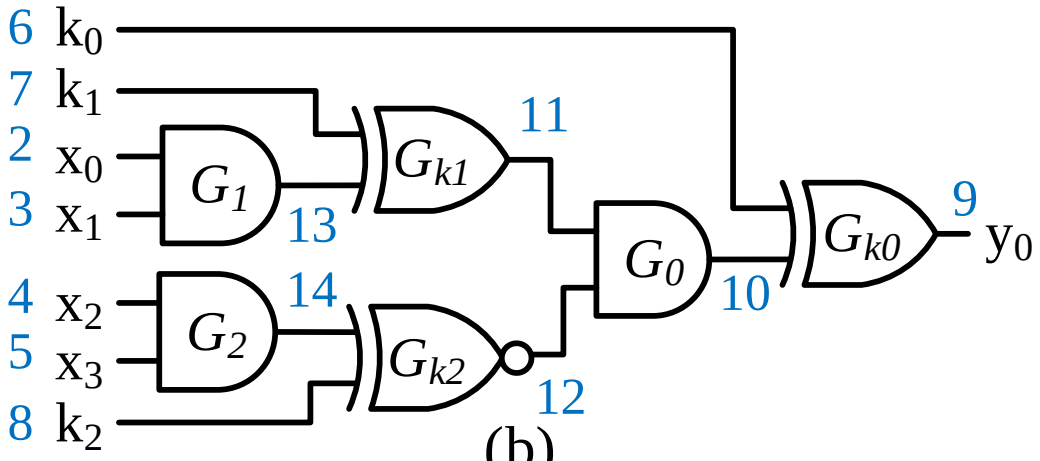
- Incorrect key elimination:
  - Half ($K_O = f(K_I)$)
  - More than half
    - 1 at AND output and 0 at OR output
  - Less than half
    - 0 at AND output and 1 at OR output



Key Pruning Analysis – Example

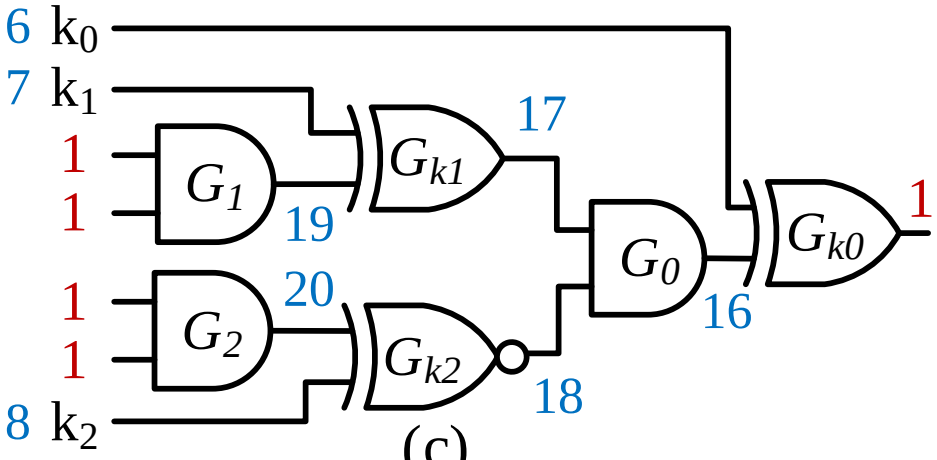


(a)

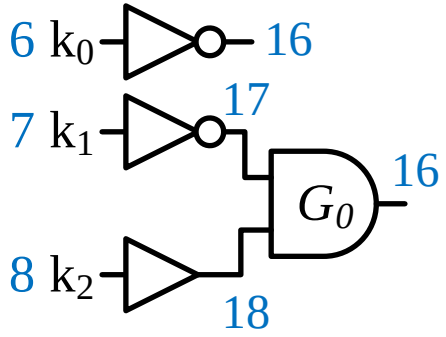


(b)

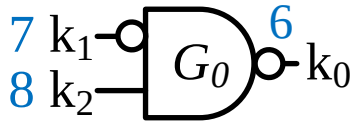
Iteration 1:



(c)



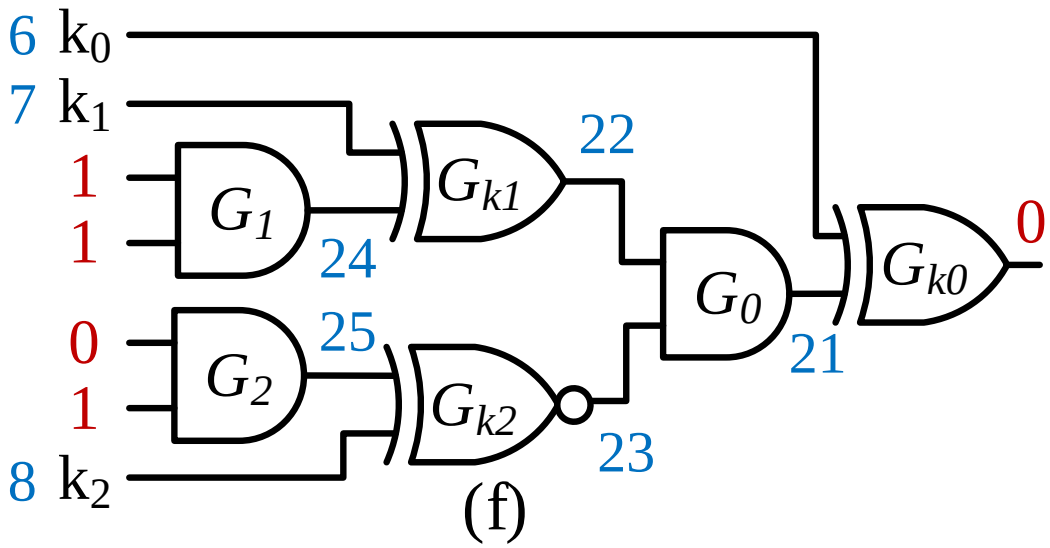
(d)



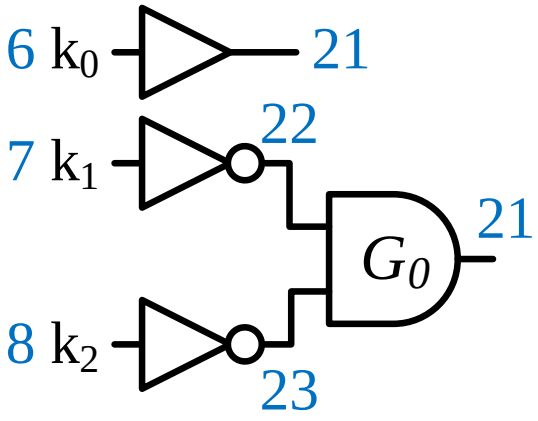
(e)

Key Pruning Analysis – Example – Cont.

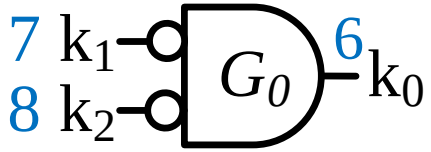
Iteration 2:



(f)



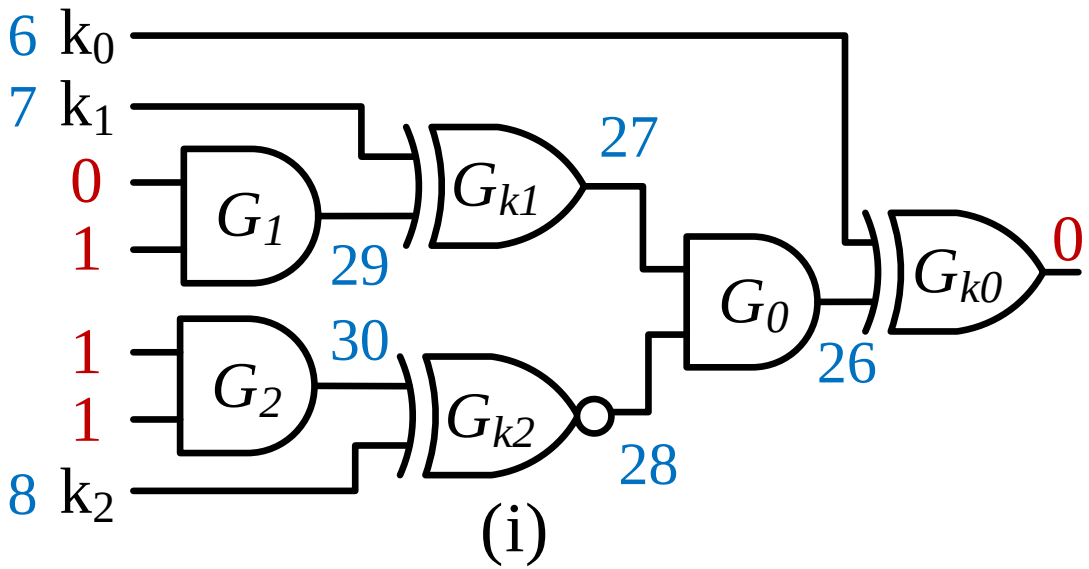
(g)



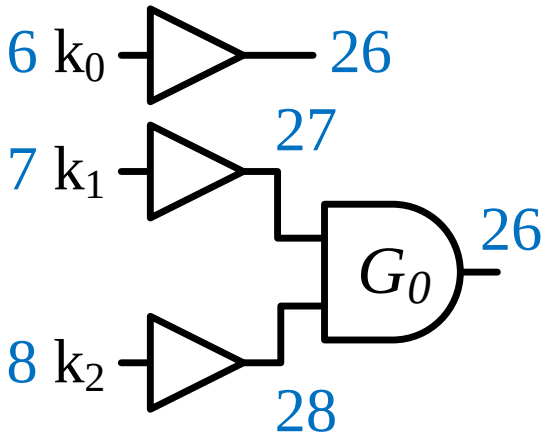
(h)

Key Pruning Analysis – Example – Cont.

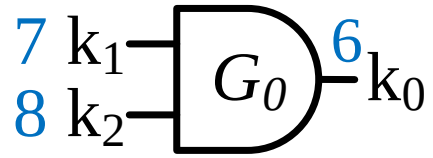
Iteration 3:



(i)



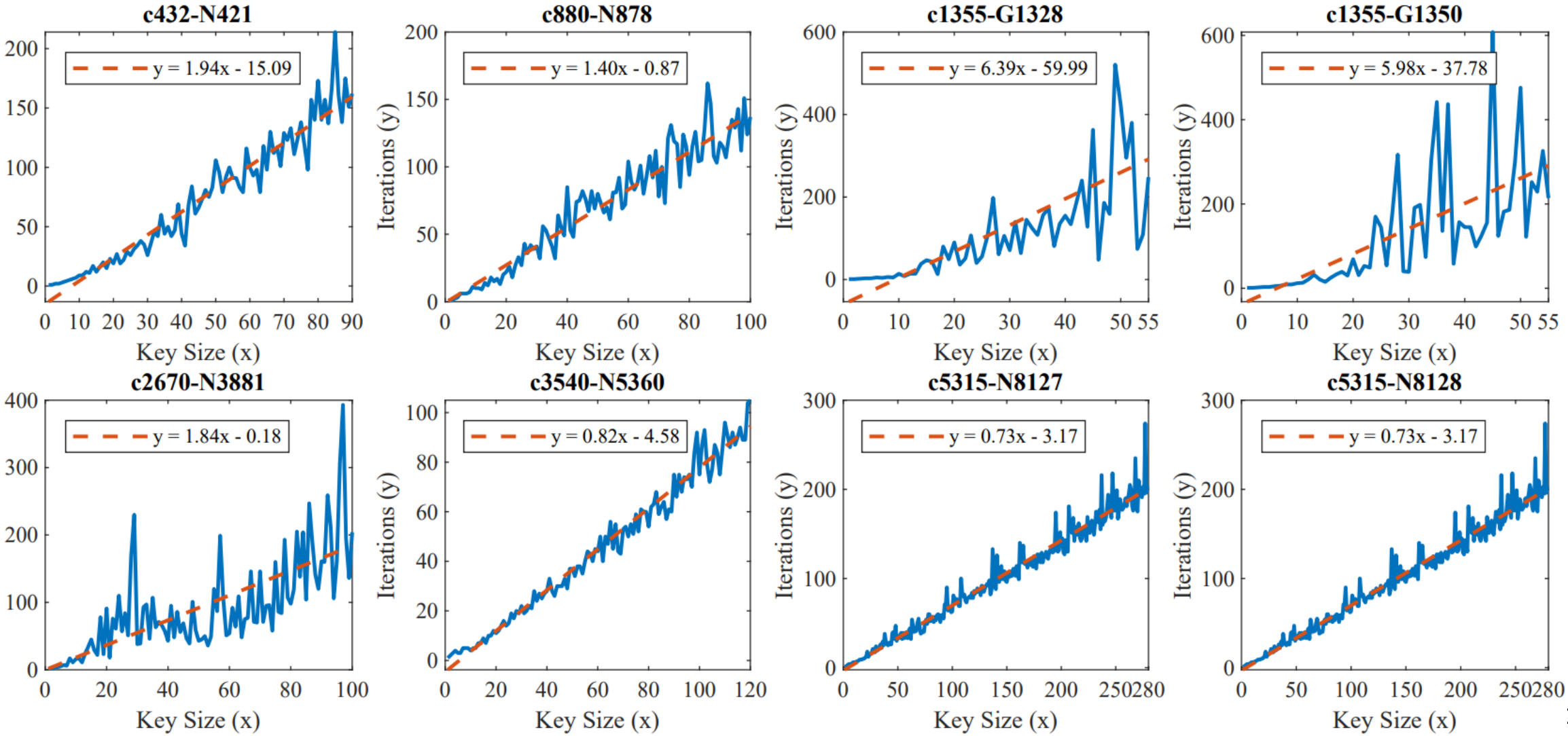
(j)



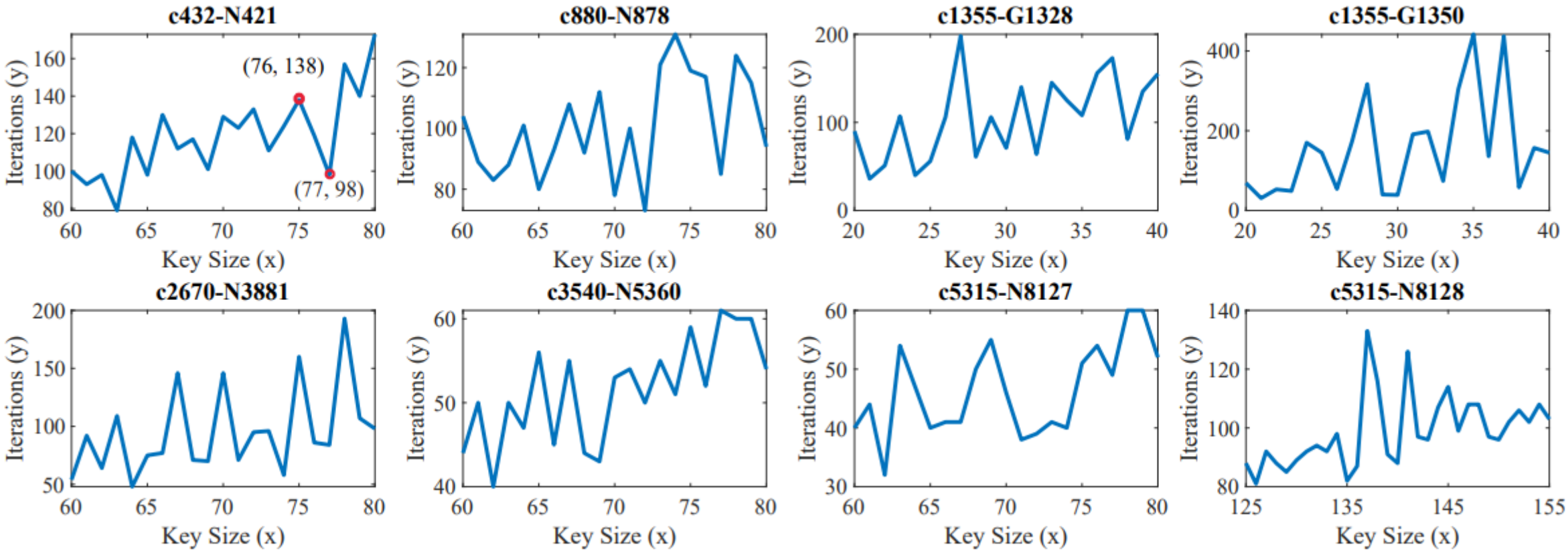
(k)

There are 3-key bits. Generally, we expect $2^3 = 8$ DIPs to break the 3-bit key. However, SAT only needs 3 DIPs.

SAT Attack Complexity Analysis

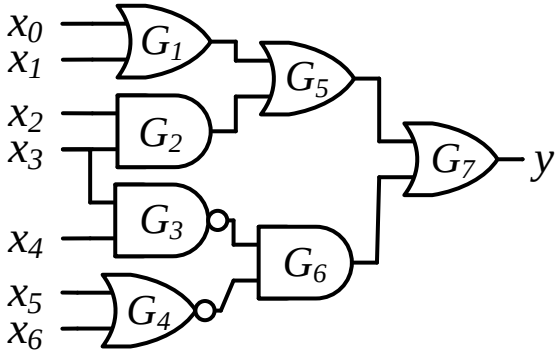


SAT Attack Complexity Analysis – Cont.

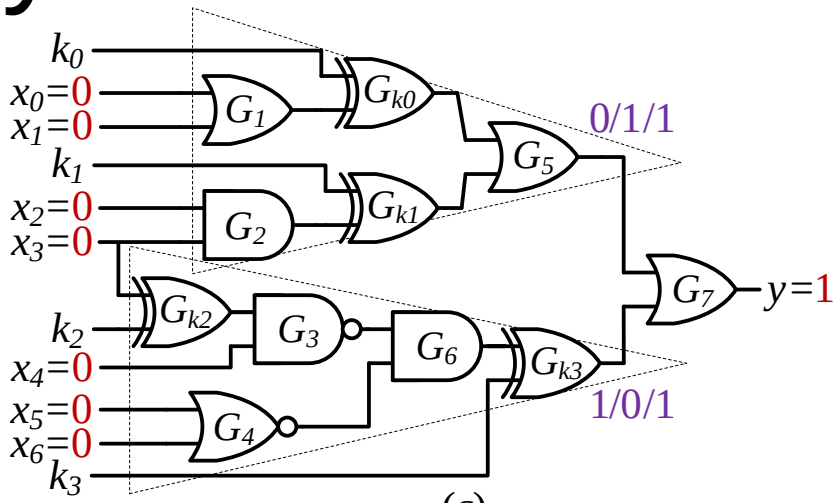


The overall SAT attack complexity is not a strict monotonically increasing function.

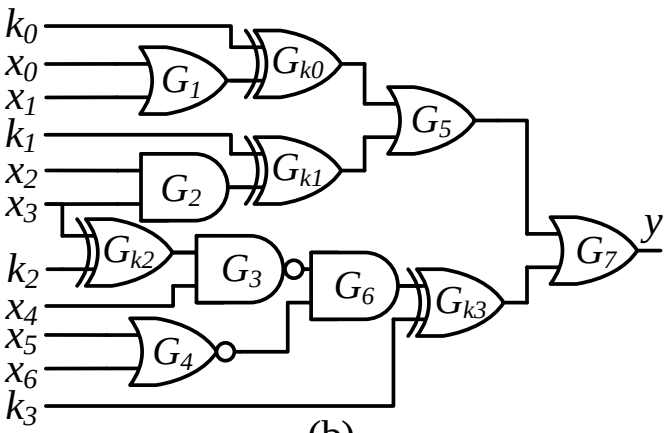
Non-Monotonically Increasing Attack Complexity



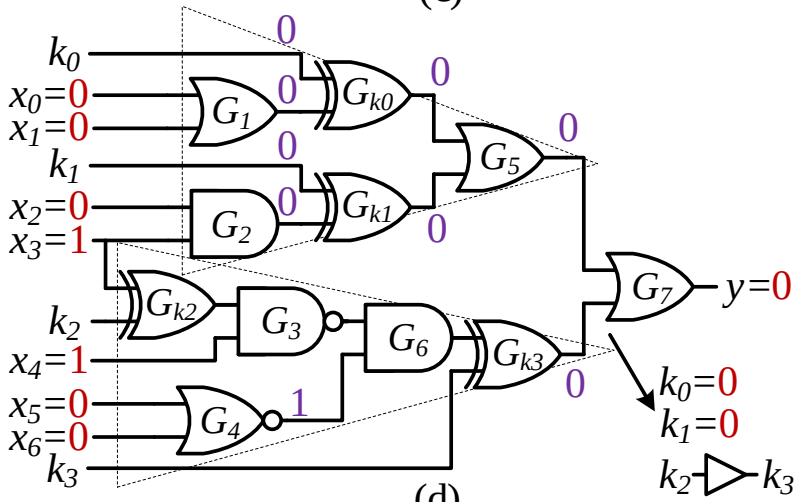
(a)



(c)



(b)



(d)

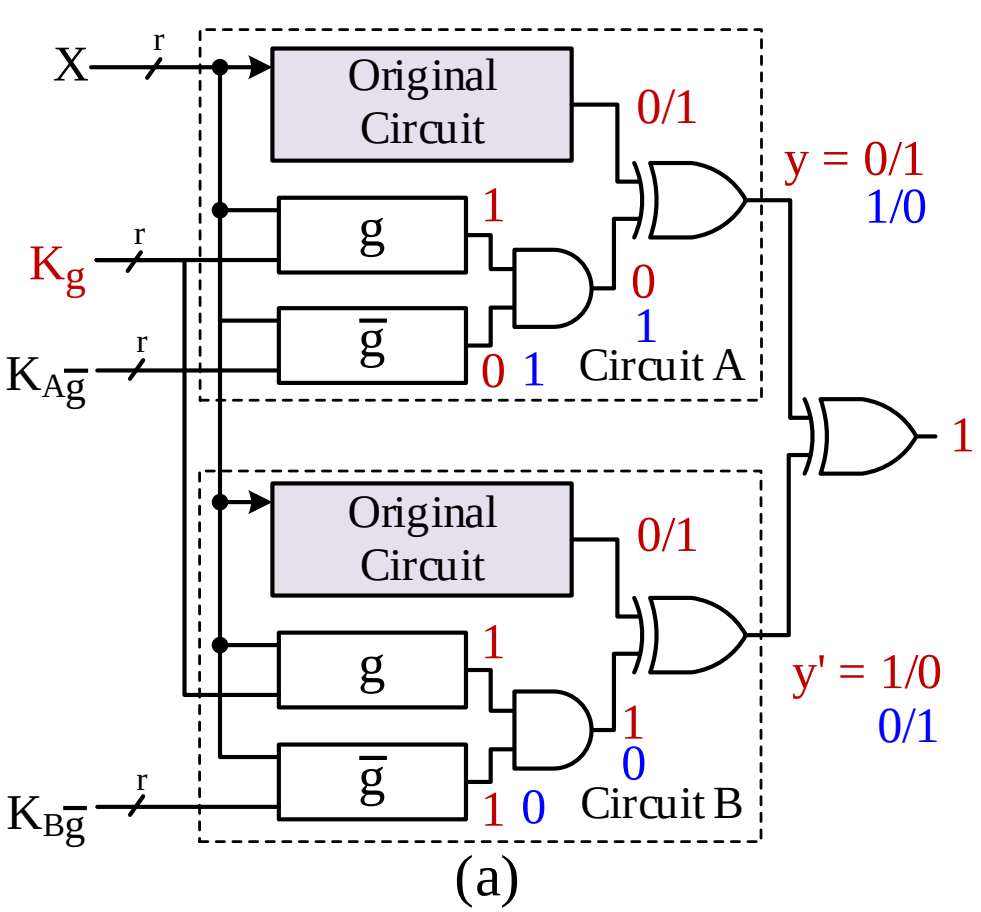
Key $\{k_0, \ldots, k_3\}$	1st IO Pair $\{0000000; 1\}$	1st IO Pair $\{0001100; 0\}$
0000	✓	✓
0001	✗	✗
0010	✓	✗
0011	✗	✓
0100	✓	✗
0101	✓	✗
0110	✓	✗
0111	✓	✗
1000	✓	✗
1001	✓	✗
1010	✓	✗
1011	✓	✗
1100	✓	✗
1101	✓	✗
1110	✓	✗
1111	✓	✗

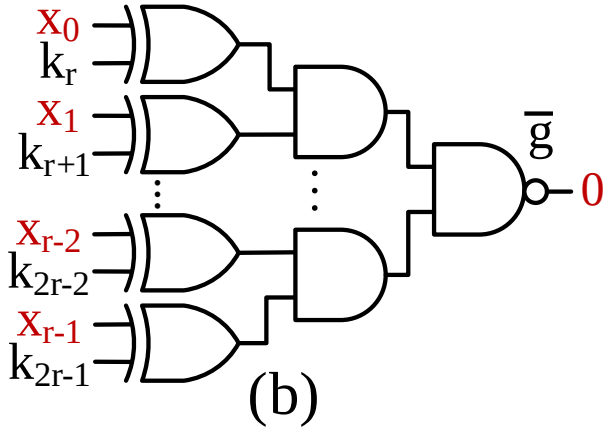
Outline

- Overview
- Background
- SAT Attack Complexity Analysis
  - Traditional XOR-based Locking
  - Point Function-based Locking
- Time complexity analysis for traditional SAT attack
- Conclusion

SAT Attack Analysis on AntiSAT

- AntiSAT block is the ANDed two key blocks, $g(X, K_A)$ and $\bar{g}(X, K_B)$.
  - g is the AND-tree for secure integration (p = 1).
  - g and $\bar{g}$ are always complementary under the correct key.
  - Only one incorrect output for any wrong key.
- Key constraint on K_g
  - 1 IO pair is sufficient to break the secret key.





$K_{\bar{g}}$

$k_r = \overline{x_0} = k_0$

$k_{r+1} = \overline{x_1} = k_1$

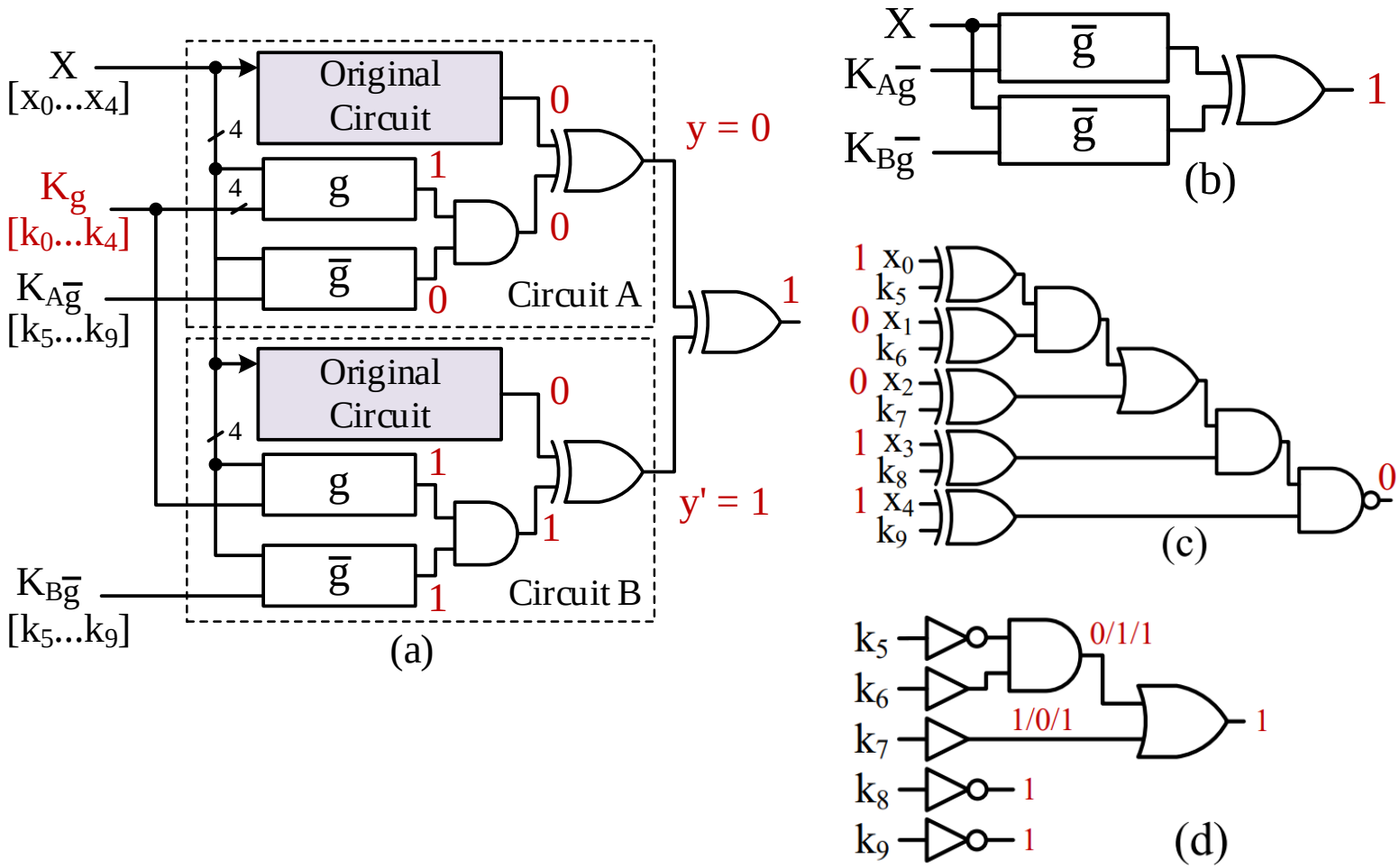
$\vdots$

$k_{2r-1} = \overline{x_{r-1}} = k_{r-1}$

(c)

SAT Attack Analysis on CAS-Lock

- CAS-Lock consists of the ANDed key blocks, $g(X, K_A)$ and $\bar{g}(X, K_B)$.
  - g is the cascaded chain of AND/OR gate.
  - g and $\bar{g}$ are always complementary under the correct key.
  - Output corruptibility is tuned by the location and number of OR gates.
- Key constraint on K_g
  - Linear number of pairs can uniquely break the secret key.



SAT Attack Analysis on TTLock and SFLL

- Oracle inside the locked circuit
  - Perturb Unit (PU) F*
  - Restore Unit (RU) G*
- Commercial synthesis tool optimization.
  - Multiple solutions for PU
  - Unique extraction for RU

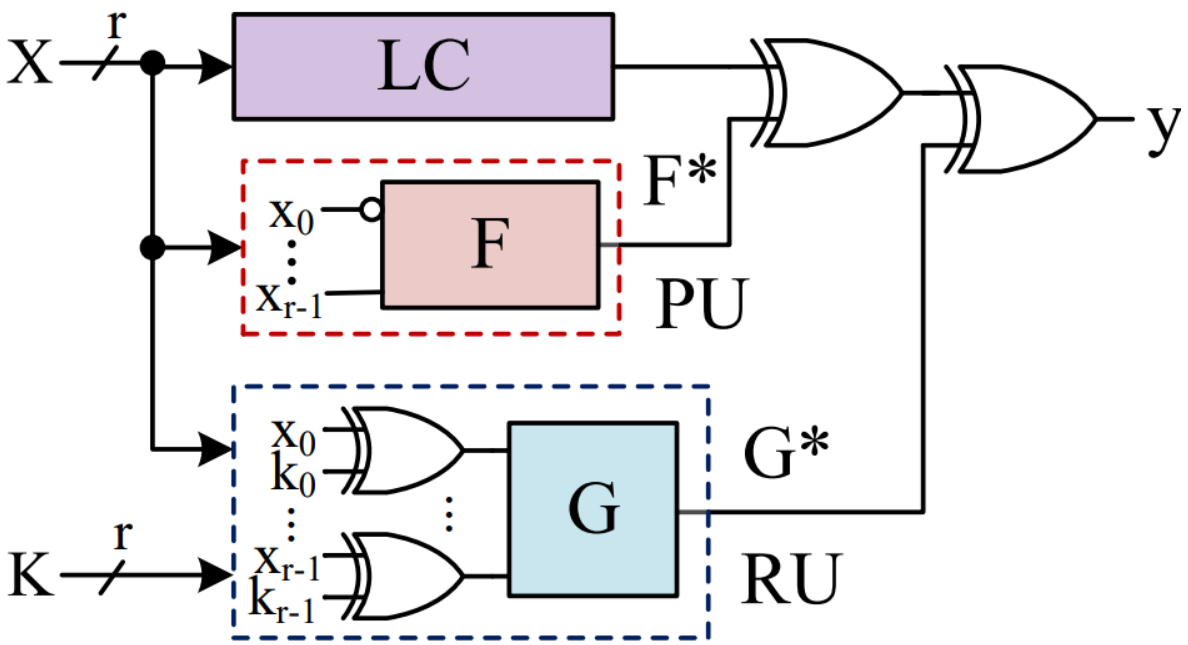


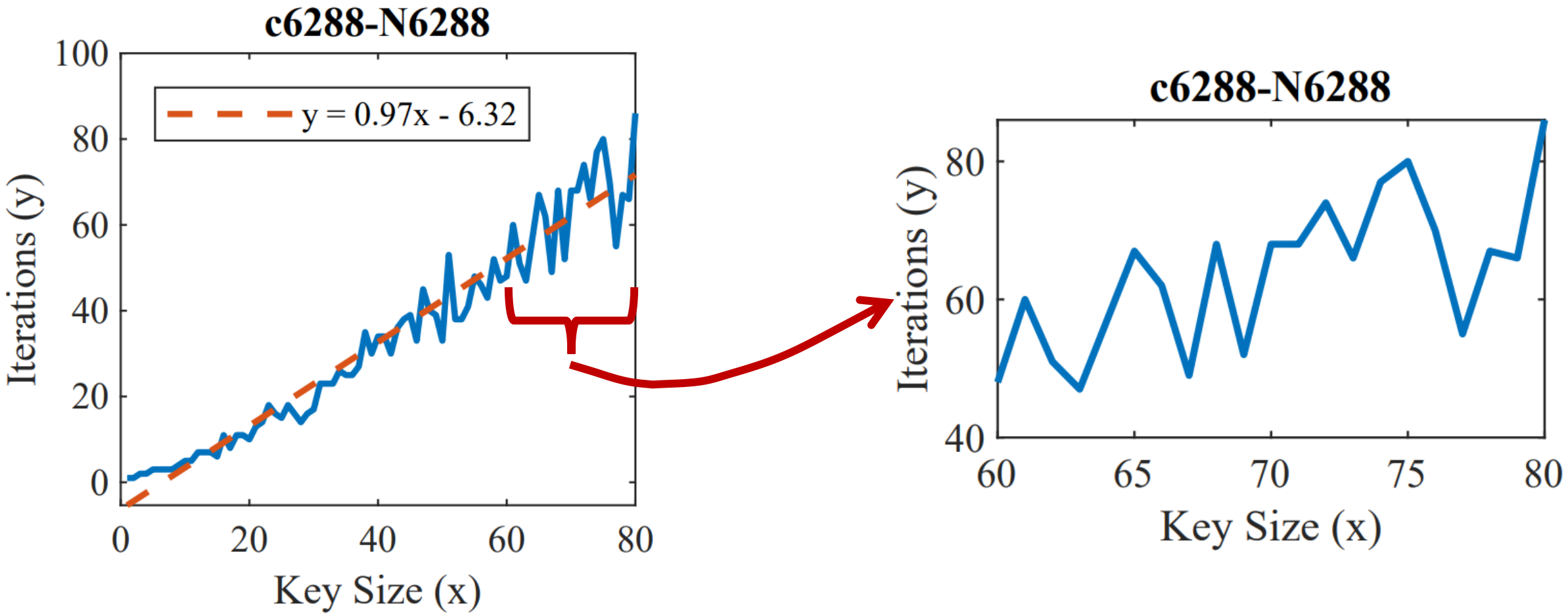
Figure: Generalized architecture of stripped functionality logic locking (SFLL). [1]

[1]. D. Sirone and P. Subramanyan, “Functional analysis attacks on logic locking,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 936–939, 2019.

Outline

- Overview
- Background
- SAT Attack Complexity Analysis
  - Traditional XOR-based Locking
  - Point Function-based Locking
- Time Complexity Analysis For Traditional SAT Attack
- Conclusion

Time Complexity Analysis For Traditional SAT Attack



Time Complexity Analysis For Traditional SAT Attack – Cont.

Algorithm 1: SAT attack on logic locking.

Input : Unlocked circuit, oracle ($C_O(X, Y)$) and locked circuit ($C(X, K, Y)$)

Output: Correct Key (K_c)

```
1  i ← 0 ;
2  F ← [];
3  while (true) do
4  |   i ← i + 1 ;
5  |   [X_i, K_i, r] = sat[F ∧ (Y_Ai ≠ Y_Bi)];
6  |   if (r == false) then
7  |   |   break;
8  |   end
9  |   Y_i = sim_eval(X_i);
10 |   F ← F ∧ C(X_i, K, Y_i);
11 end
12 K_c ← K_i;
13 return K_c ;
```

Start Time

End Time

$\|K\|$	$\|P\|$	CPU time (s)				$\frac{UNSAT}{Total}$ (%)
		Total	IO Pairs	Average	UNSAT	
1	1	86.351	0.09108	0.09108	86.25948	99.894
5	4	79.614	0.12705	0.03176	79.48717	99.840
10	7	62.018	0.14788	0.02113	61.87004	99.761
15	10	77.133	0.17205	0.01721	76.96051	99.776
20	14	92.588	0.32586	0.02328	92.26268	99.648
25	20	88.295	2.48402	0.12420	85.81125	97.186
30	16	73.065	0.84954	0.05310	72.21507	98.837
35	29	86.737	14.53748	0.50129	72.19920	83.239
40	27	149.097	13.34636	0.49431	135.7502	91.048
45	41	1130.466	18.31241	0.44664	1112.154	98.380
50	37	84.404	6.16717	0.16668	78.23738	92.693
55	45	1188.844	57.14645	1.26992	1131.698	95.193

Conclusion

- SAT attack complexity for logic locking is linear in iterations due to the removal of large number of incorrect keys per iteration.
- This is the first time a non-monotonically increase in SAT complexity under increased key sizes is reported.
- We give analytical reasoning for SAT attack on post-SAT solutions, AntiSAT, CAS-Lock, TTLock and SFLL.
- For future SAT-resilient locking schemes, one can target on achieving the same degree of time complexity like the last iteration of c6288 benchmark.

Any Questions?

Thank you!

Contact:

Prof. Ujjwal Guin, Auburn University
Email: ujjwal.guin@auburn.edu