Mitigating SAT Attack on Logic Locking

Yang Xie and Ankur Srivastava

University of Maryland

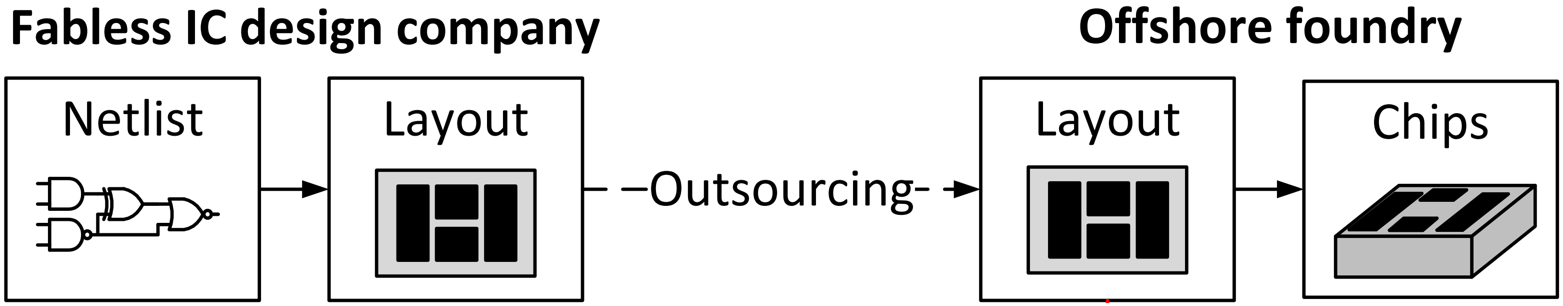
CHES 2016

Outline

- Security threats in IC fabrication outsourcing
- Logic Locking
- SAT Attack
- Anti-SAT Block Design
- Results
- Conclusion

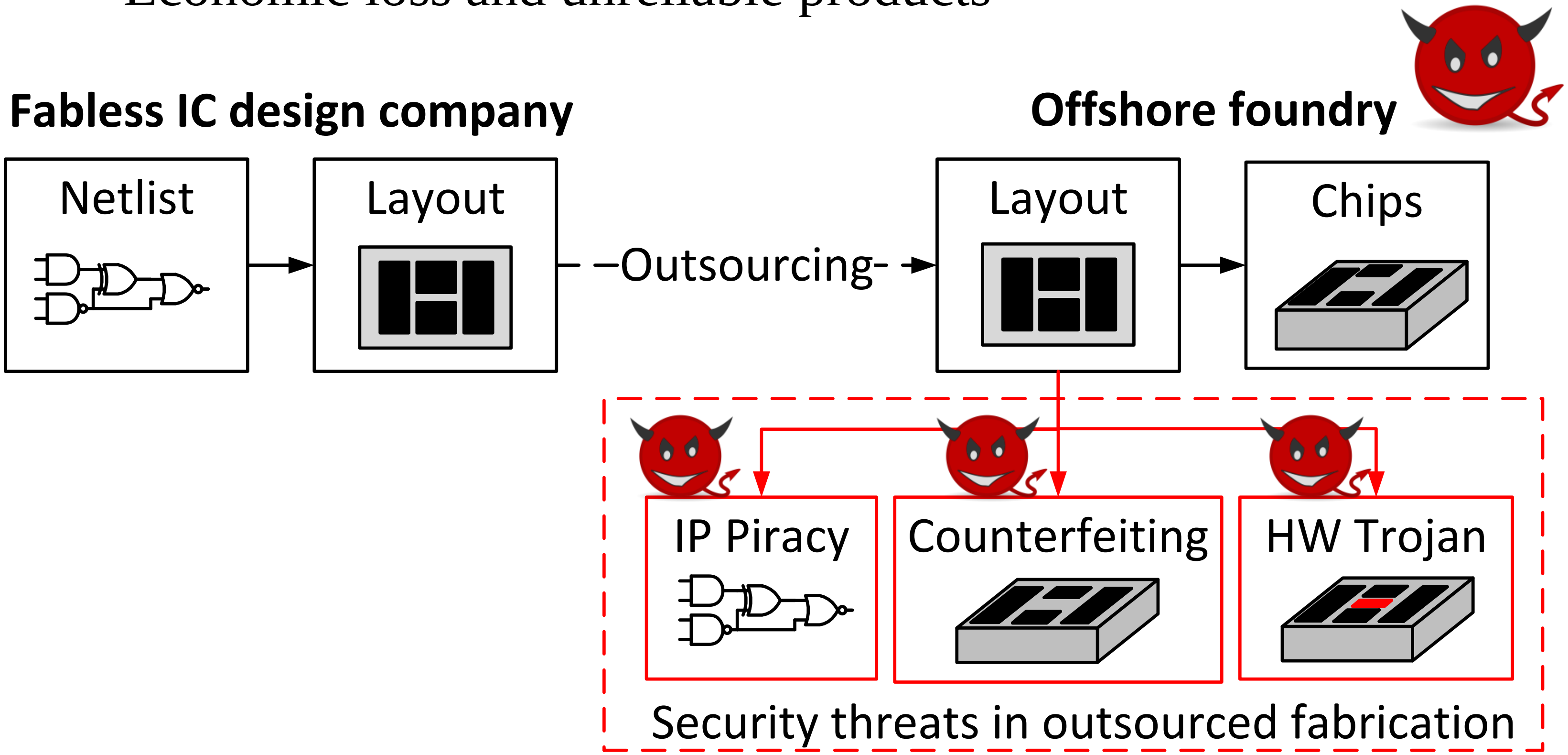
Supply Chain Security

- **IC fabrication outsourcing**
  - Semiconductor fab is expensive (> $15 billion by 2020 [1]).
  - Increasing complexity of IC designs



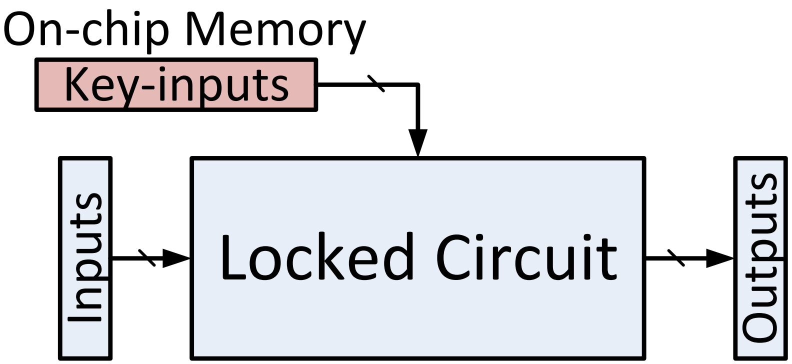
Supply Chain Security

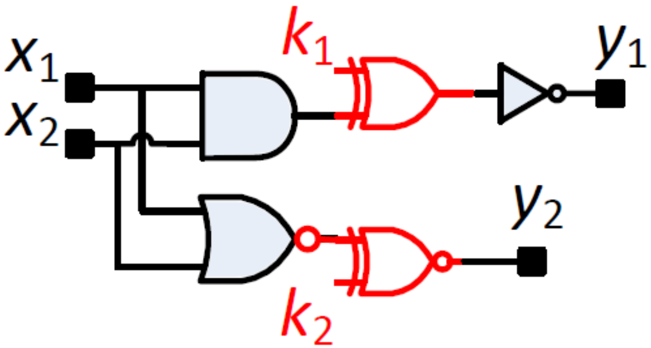
- **The foundry might not be trustworthy**
  - IP Piracy, counterfeiting, hardware Trojan insertion…
  - Economic loss and unreliable products



Logic Locking

- **Logic Locking* [2-13]:**
  - During fabrication time, the designer locks the circuit by adding additional logic gates (***key-gates***) and ***key-inputs***
  - The locked circuit preserves the original functionality only when a correct key is loaded into the on-chip memory

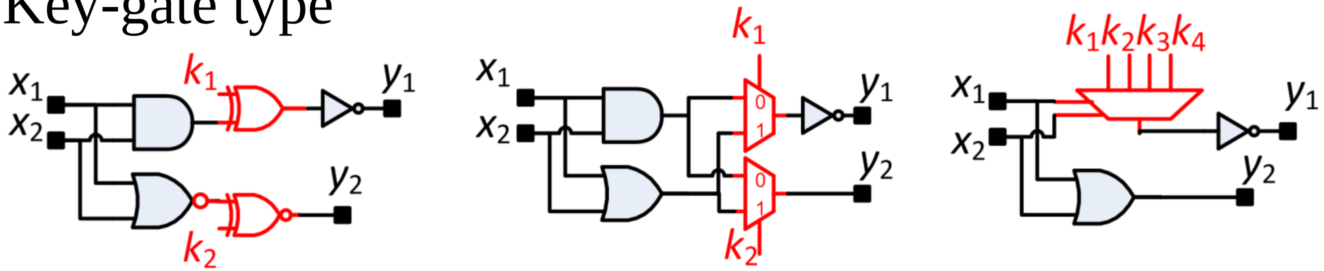




*Some literatures called it logic obfuscation and logic encryption

Logic Locking

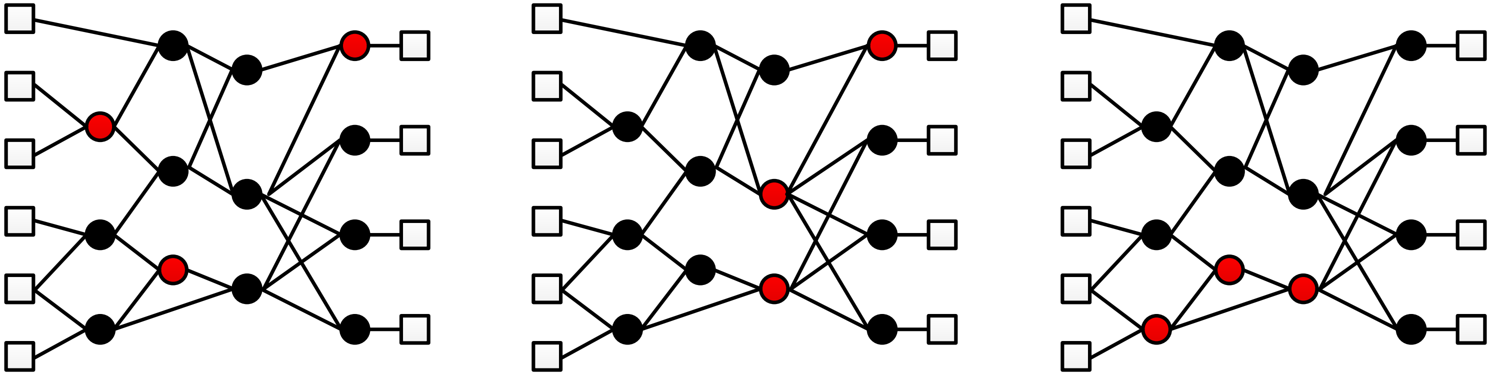
- **Various logic locking techniques [2-13]:**
  - Key-gate type



XOR/XNOR-based MUX-based LUT-based

  - Key-gate insertion algorithms

Goals: (1) increase output corruptibility and (2) prevent key-learning



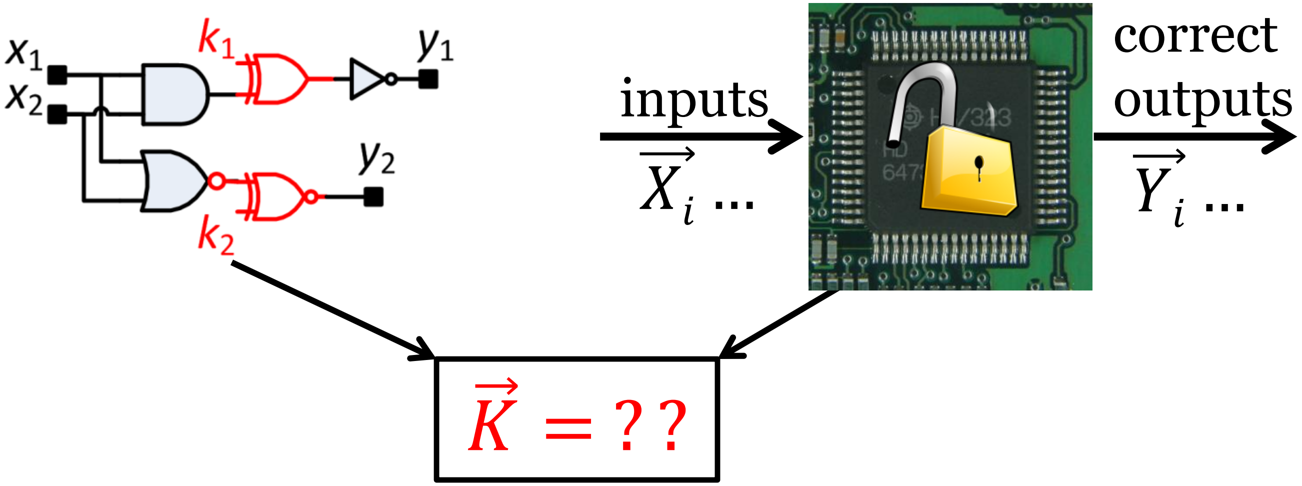
Random Fault analysis Interference analysis

Attacks on Logic Locking

- **Attack model [7, 8,11,13]:**
  - Goal: obtain the correct key
  - Knowledge:

    1) ***A locked netlist*** obtained by reverse-engineering the layout

    2) ***An activated chip*** obtained from open market, which can be used to observe correct I/O pairs as *a black box*



inputs

$\vec{X}_i$ ...

correct outputs

$\vec{Y}_i$ ...

$\vec{K} = ??$

Attacks on Logic Locking

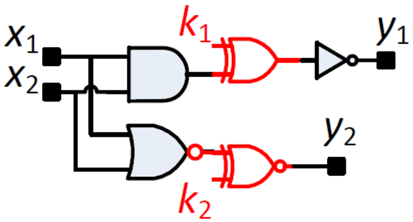
- **Key search based attack** [7,8,13]
  - Test the correctness of a key using a subset of correct I/O pairs
  - Does not guarantee a successful attack especially when
    - key-size is large(e.g. >128) [7]
    - key-gate types and locations are carefully selected [7,8,13]
  - The obtained key is only “correct” w.r.t. that subset of I/O pairs.
- ***SAT based attack*** [11]
  - ***Theoretically sound***: guarantees to obtain the correct key w.r.t. all I/O pairs upon termination
  - ***Efficient***: break most logic locking techniques proposed in [5,6,10,11,12] within a few hours even for a reasonably large number of keys (e.g. >1000).

SAT Attack Algorithm

- **Basic idea**
  - To ***iteratively*** find a set of ***special inputs*** and observe their ***outputs*** till they can identify all the ***wrong key combinations***
  - Formulated as ***SAT formulas*** and solved by SAT solvers

Def. 1 Wrong key combinations (WK):

- Example:

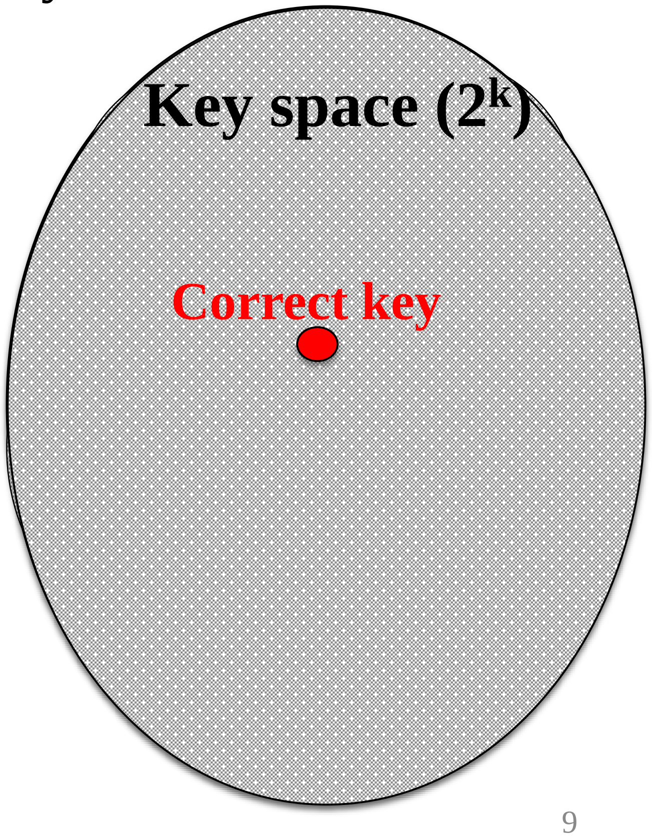


$$(\vec{X},\vec{Y}) = (11,01)$$

$$\Rightarrow \vec{K} \neq (01,10,11)$$

Def. 2 Distinguishing I/O pair (DIP)

- An I/O pair at *i*-th iteration is a DIP if it can identify a "*unique*" subset of wrong key combinations that cannot be identified by previous *i-1* DIPs.

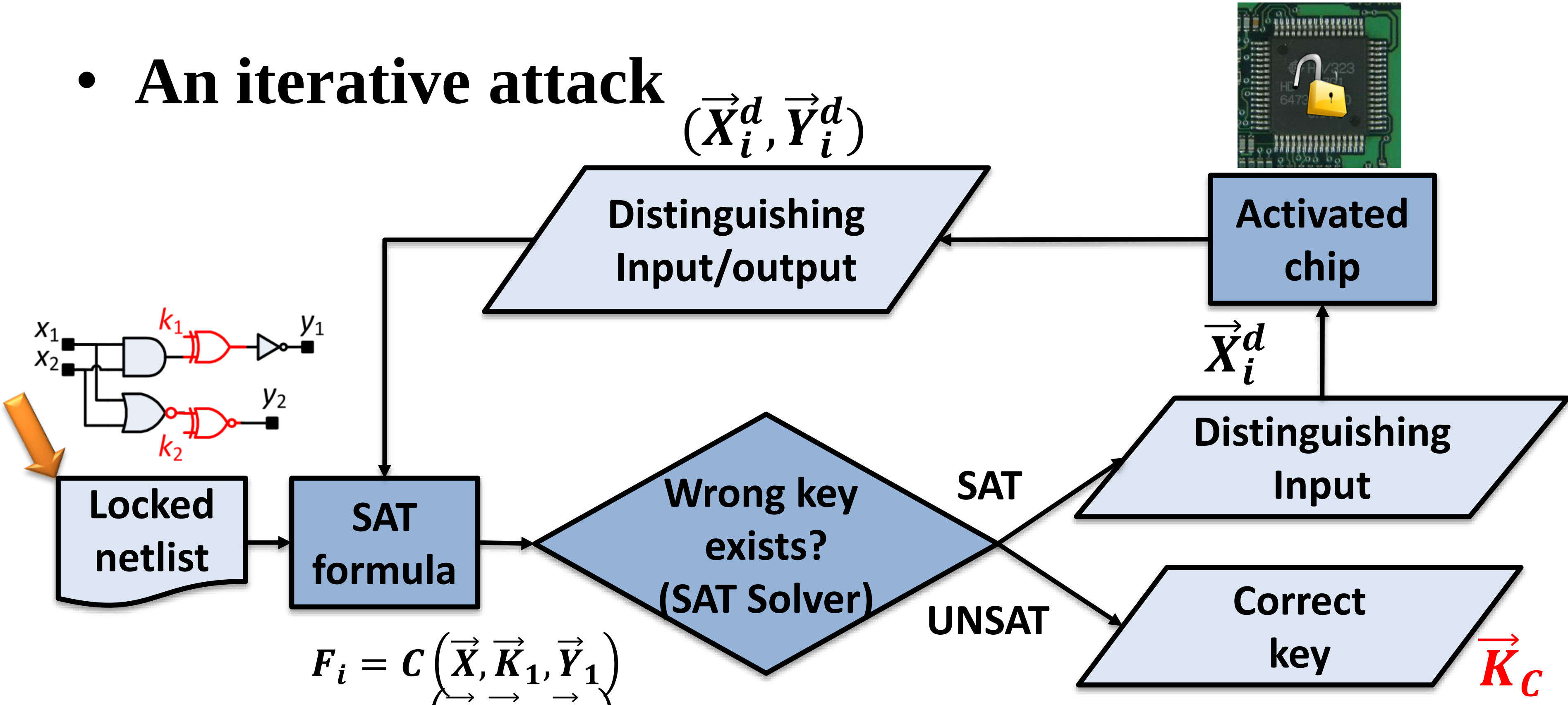


Key space (2^k)

Correct key

SAT Attack Algorithm

- **An iterative attack**



Locked netlist

SAT formula

Wrong key exists? (SAT Solver)

Distinguishing Input/output

Activated chip

Distinguishing Input

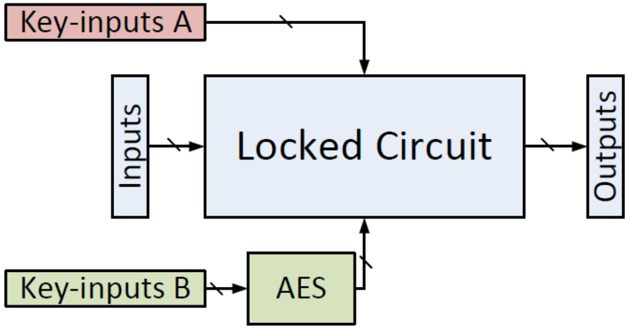
Correct key

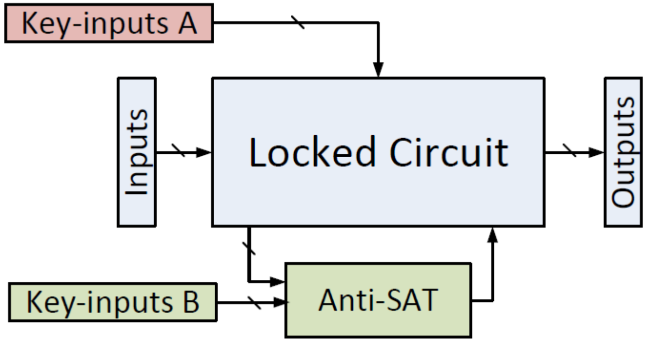
$$
\begin{aligned}
F_i = {} & C\left(\vec{X}, \vec{K}_1, \vec{Y}_1\right) \\
& \wedge\ C\left(\vec{X}, \vec{K}_2, \vec{Y}_2\right) \\
& \wedge\ (\vec{Y}_1 \neq \vec{Y}_2) \\
& \wedge\ C\left(\vec{X}_i^d, \vec{K}_1, \vec{Y}_i^d\right) \\
& \wedge\ C\left(\vec{X}_i^d, \vec{K}_2, \vec{Y}_i^d\right) \\
& \wedge\ \quad \ldots
\end{aligned}
$$

Efficient because ***the number of iterations (# DIPs) required to identify all the wrong keys is small*** [13]

SAT Attack Efficiency Analysis

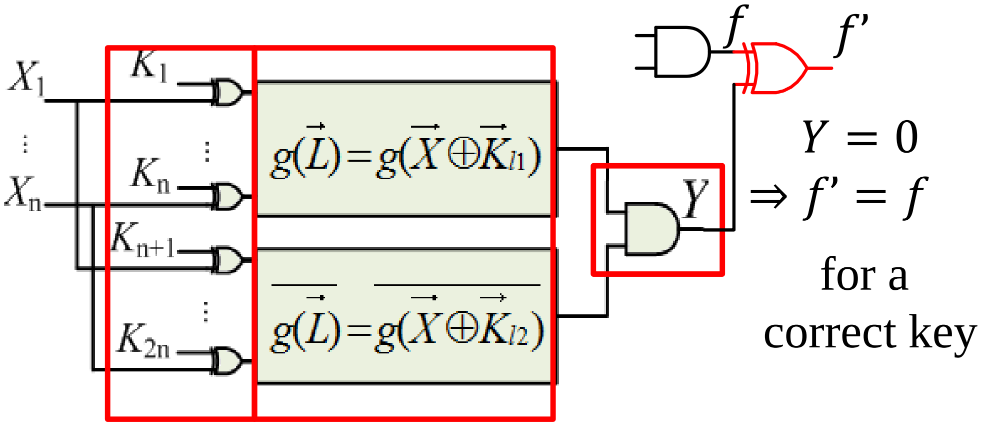
- **Total execution time** $T = \sum_{i=1}^{\lambda} t_i$
  - $\boldsymbol{t_i}$: **SAT solving time for *i*-th iteration**
    - Depends on benchmark characteristics (hard-SAT circuits like Multiplier)
    - Idea: add an AES to increase the SAT solving time [4]
    - Drawback: significant overhead
  - $\boldsymbol{\lambda}$: **total number of iterations**
    - Depends on key-size and key-gate location. However, previous logic locking cannot effectively counter SAT attack
    - Idea: add our proposed ***Anti-SAT block*** such that ***λ is exponential to the key-size***





Anti-SAT Block

- **An *n*-input Anti-SAT block**
  - Two *n*-input logic blocks $g\left(\vec{L}\right)$ and $\overline{g\left(\vec{L}\right)}$
  - *2n* key-gates (XOR or XNOR) at their inputs
  - Outputs of two logic blocks are fed into an AND gate


X_1
X_n
K_1
K_n
K_{n+1}
K_{2n}
$g(\vec{L})=g(\vec{X}\oplus\vec{K}_{l1})$
$\overline{g(\vec{L})}=\overline{g(\vec{X}\oplus\vec{K}_{l2})}$
Y
f
f'

$Y = 0$
$\Rightarrow f' = f$
for a
correct key

$\vec{L}$	$g(\vec{L})$	$\overline{g(\vec{L})}$
000…000	0	1
000…001	0	1
⋮	⋮	⋮
111…111	1	0

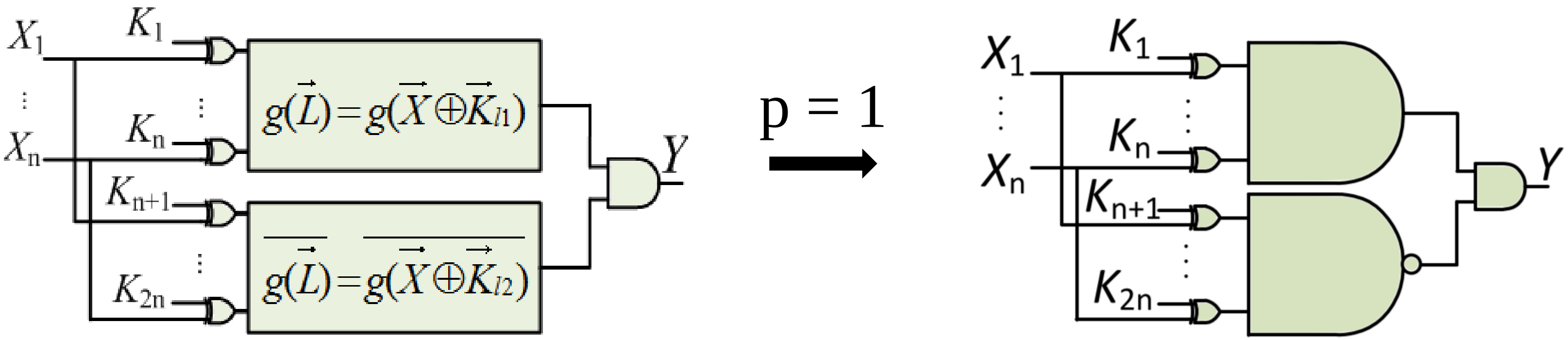
- **Constant-output property**
  - For a correct key, the output of the Anti-SAT block is always 0
  - For an incorrect key, the output can be 0 (correct) or 1 (incorrect)

Security Analysis of Anti-SAT Block

- **Theorem 1:** Assuming the output-one count p of the n-input function $g(\vec{L})$ is sufficiently close to 1 or sufficiently close to 2^n-1, the number of iterations λ needed by the SAT attack to decipher the correct key is lower bounded by 2^n.
- **Sketch of the proof:**
  1) Assuming there exists p input vectors that make $g(\vec{L})$ outputs one (so $2^n - p$ input vectors that make $\overline{g(\vec{L})}$ output one).
  2) Show that each iteration can identify $\leq \boldsymbol{p\cdot(2^n-p)}$ *unique* wrong key combinations.
  3) Show that total #wrong key combinations = $(\boldsymbol{2^{2n}-2^n})$.
  4) Show that it needs $\boldsymbol{\lambda} \geq \frac{\boldsymbol{2^{2n}-2^n}}{\boldsymbol{p\cdot(2^n-p)}}$ iterations to identify all wrong keys.
  5) When $p \rightarrow 1$ or $p \rightarrow 2^n-1$, we have $\boldsymbol{\lambda \geq 2^n}$. Hence proved.

Security Analysis of Anti-SAT Block

- **Theorem 1:** Assuming the output-one count p of the n-input function $g(\vec{L})$ is sufficiently close to 1 or sufficiently close to $2^n - 1$, the number of iterations λ needed by the SAT attack to decipher the correct key is lower bounded by 2^n.

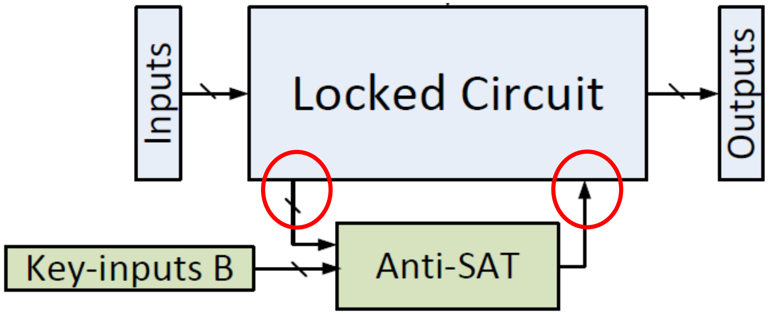


X_1
X_n
K_1
K_n
K_{n+1}
K_{2n}
$g(\vec{L}) = g(\vec{X} \oplus \vec{K}_{l1})$
$\overline{g(\vec{L})} = \overline{g(\vec{X} \oplus \vec{K}_{l2})}$
Y
X_1
X_n
K_1
K_n
K_{n+1}
K_{2n}
Y

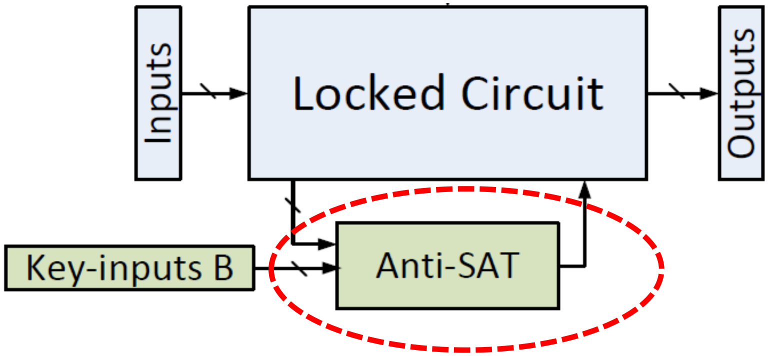
p = 1

Anti-SAT Block

- **How to integrate the Anti-SAT block?**
- **How to prevent removal attack?**



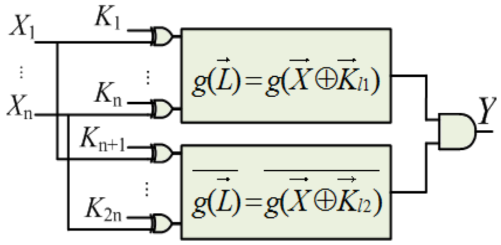
Input and output locations

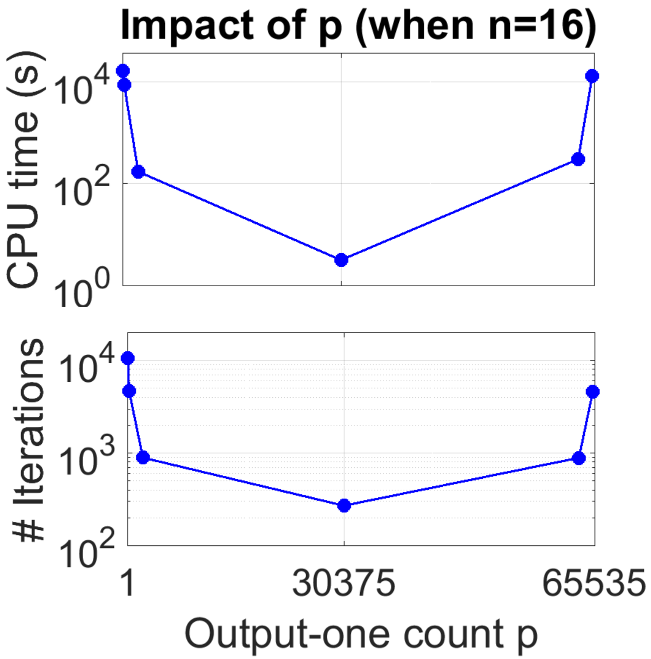


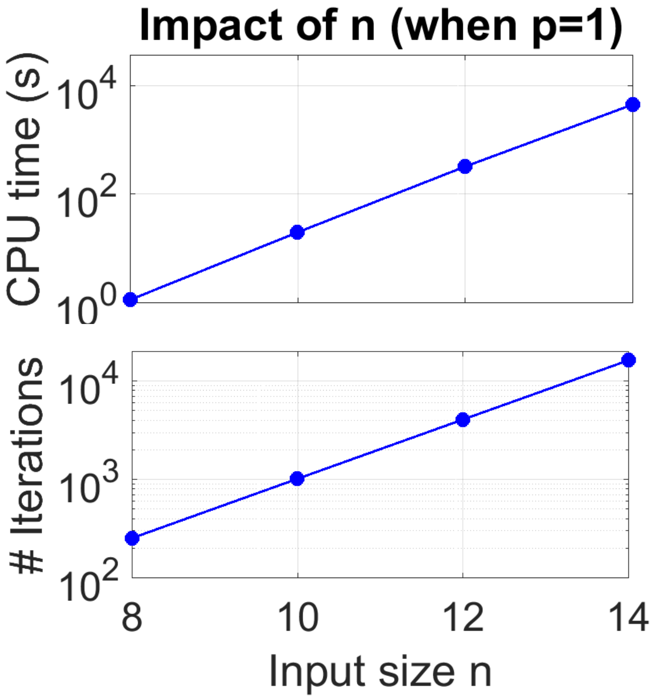
Min-cut partitioning based removal attack

SAT Attack Results

- **Anti-SAT block design**
  - Relationship between λ, n, p: $\lambda \geq \frac{2^{2n}-2^n}{p\cdot(2^n-p)}$
  - When $p \rightarrow 1$ or $p \rightarrow 2^n - 1$, we have $\lambda \rightarrow 2^n$


X_1
X_n
K_1
K_n
K_{n+1}
K_{2n}
$g(\vec{L})=g(\vec{X}\oplus\vec{K}_{l1})$
$\overline{g(\vec{L})}=\overline{g(\vec{X}\oplus\vec{K}_{l2})}$
Y


Impact of p (when n=16)

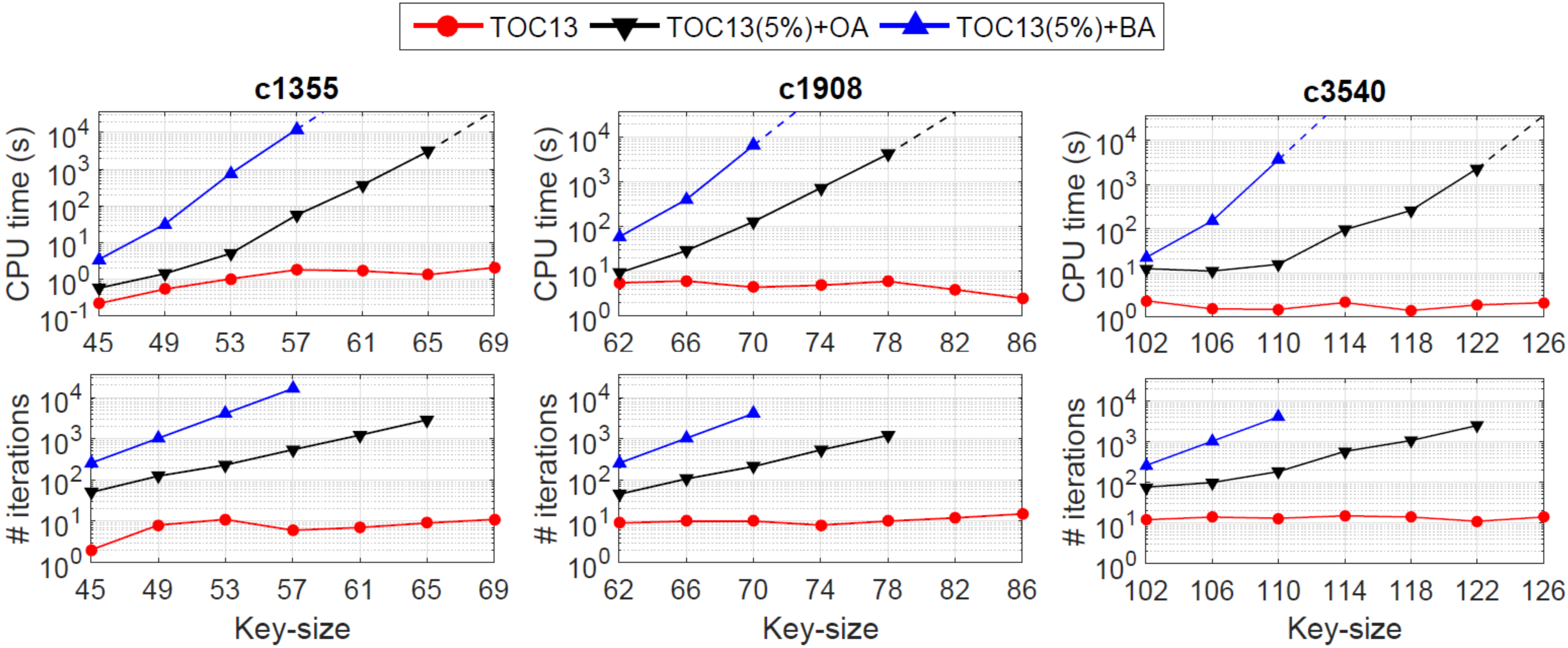

Impact of n (when p=1)

SAT Attack Results

- **Anti-SAT block application**
  - 6 benchmarks for ISCAS85 and MCNC (500+ ~ 6000+ gates)
  - Three setups:
    - TOC13: insert XOR/XNOR gates at the original netlist to increase output corruptibility
    - TOC13 (5%) + *n*-bit baseline Anti-SAT (*n*-bit BA)
    - TOC13 (5%) + *n*-bit obfuscated Anti-SAT (*n*-bit OA)

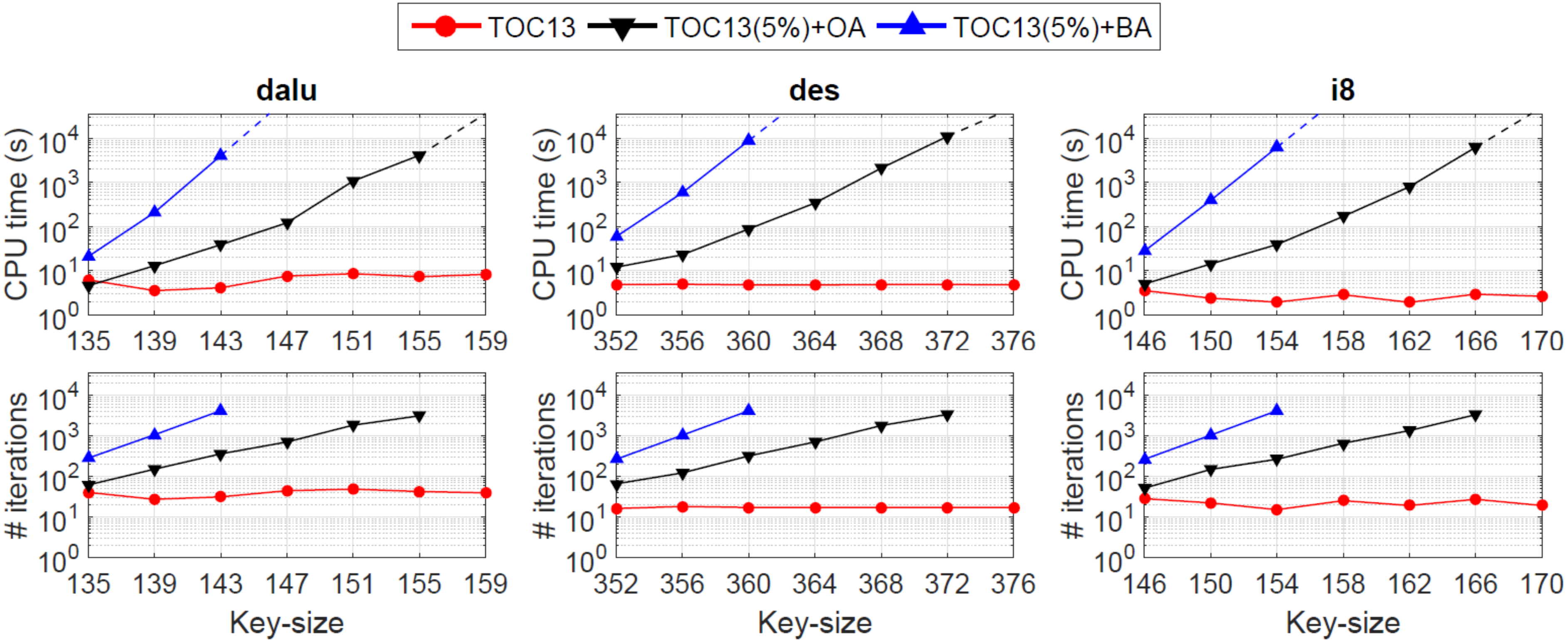
SAT Attack Results

- **Anti-SAT block application (part 1)**



SAT Attack Results

- **Anti-SAT block application (part 2)**



TOC 13 only: unlocked in 48 iterations and 8.48 seconds

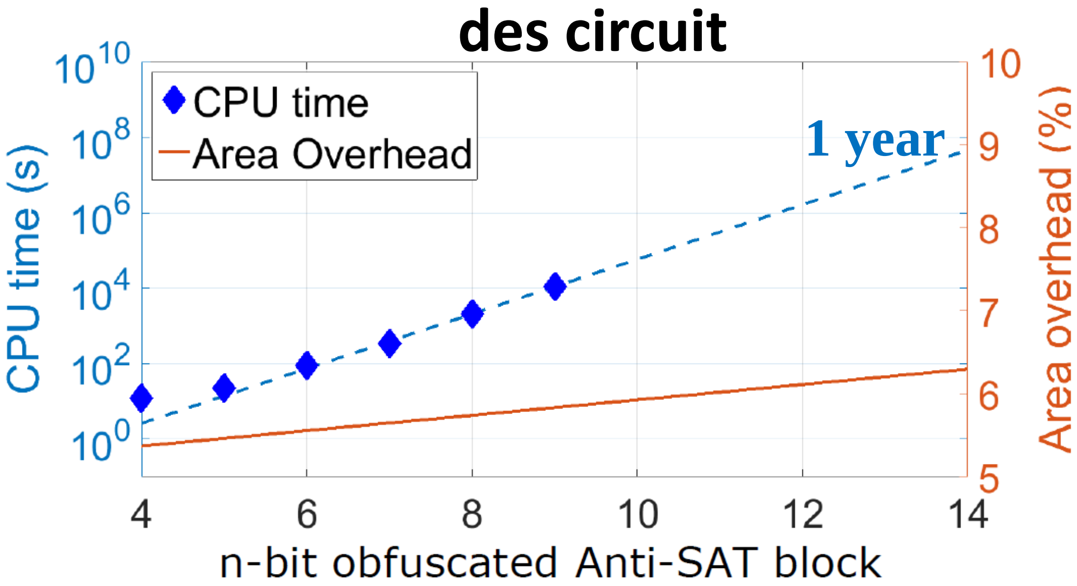
TOC 13 (5%) + n-bit BA: SAT-attack timeouts when $k_{BA} = 28$

TOC 13 (5%) + n-bit OA: SAT-attack timeouts when $k_{OA} = 40$

Overhead Result

- **Performance overhead**

des circuit



  - A *linear* increase in area overhead can result in *exponential* increase in SAT attack's computation complexity
  - TOC13 +14-bit OA (~7% overhead) can result in 1 year SAT attack time (extrapolated)

Conclusion

- A circuit block called *Anti-SAT* was proposed to mitigate the SAT attack on logic locking.
- We showed (using a rigorous mathematical proof) that the *#iterations* required by the SAT attack to reveal the correct key is *exponential to the key-size* of the Anti-SAT block.
- The Anti-SAT block was integrated to the circuit to defend SAT attack. Several *obfuscation techniques* were proposed to make the Anti-SAT block less distinguishable in order to defend the removal attack.
- Experiments results validated that a *linear increase* in performance *overhead* can result in *exponential increase* in SAT attack's *computation complexity*.

Thank you! Questions?

Mitigating SAT Attack on Logic Locking

Yang Xie and Ankur Srivastava

University of Maryland

CHES 2016

References

[1] Gartner Inc. “Market Trends: Rising Costs of Production Limit Availability of Leading-Edge Fabs.” [Online]. Available: https://www.gartner.com/doc/2163515, 2012.

[2] Yasin, Muhammad, et al. "On improving the security of logic locking.”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2015).

[3] Baumgarten, A., Tyagi, A., Zambreno, J. “Preventing IC piracy using reconfigurable logic barriers.” IEEE Design & Test of Computers (2010)

[4] Dupuis, Sophie, et al. "A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans." 2014 IEEE 20th International On-Line Testing Symposium (IOLTS). IEEE, 2014.

[5] Khaleghi, Soroush, Kai Da Zhao, and Wenjing Rao. "IC piracy prevention via design withholding and entanglement." The 20th Asia and South Pacific Design Automation Conference. IEEE, 2015.

[6] Liu, Bao, and Brandon Wang. "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks." Proceedings of the conference on Design, Automation & Test in Europe. European Design and Automation Association, 2014.

[7] Plaza, Stephen M., and Igor L. Markov. "Solving the third-shift problem in IC piracy with test-aware logic locking." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34.6 (2015): 961-971.

[8] Rajendran, Jeyavijayan, et al. "Security analysis of logic obfuscation." Proceedings of the 49th Annual Design Automation Conference. ACM, 2012.

[9] Rajendran, Jeyavijayan, et al. "Fault analysis-based logic encryption." IEEE Transactions on Computers 64.2 (2015): 410-424.

[10] Roy, Jarrod A., Farinaz Koushanfar, and Igor L. Markov. "EPIC: Ending piracy of integrated circuits." Proceedings of the conference on Design, automation and test in Europe. ACM, 2008.

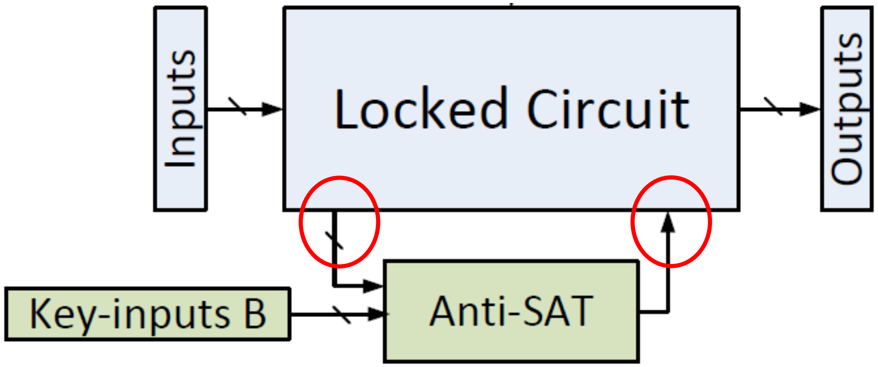
[11] Subramanyan, Pramod, Sayak Ray, and Sharad Malik. "Evaluating the security of logic encryption algorithms." Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on. IEEE, 2015.

[12] Wendt, James B., and Miodrag Potkonjak. "Hardware obfuscation using PUF-based logic." Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, 2014.

[13] Lee, Yu-Wei, and Nur A. Touba. "Improving logic obfuscation via logic cone analysis." 2015 16th Latin-American Test Symposium (LATS). IEEE, 2015.

Anti-SAT Block

- **Anti-SAT block location**
  - **Need to ensure that the # iterations (# DIPs) is still large**
  - **Input locations:** shall be connected to original wires that are highly independent
  - **Output location:** shall be connected to original wire that has high observability from the primary outputs

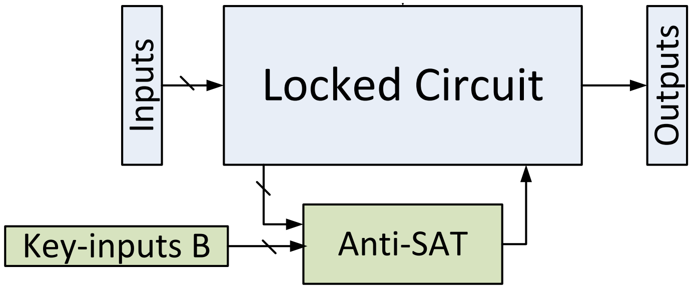


Anti-SAT Block

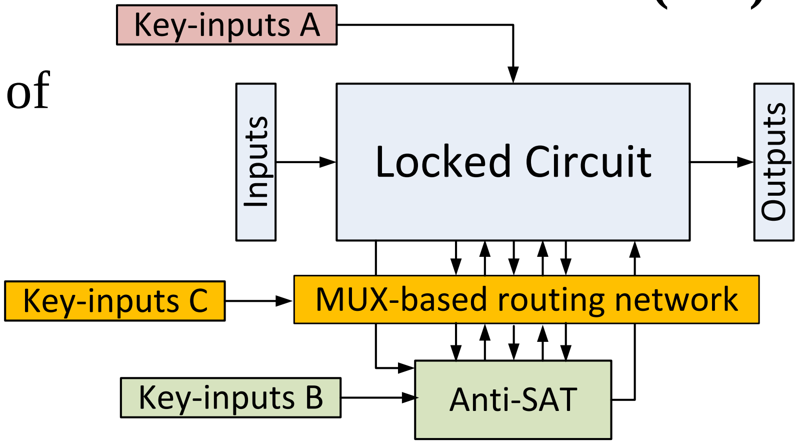
- **Anti-SAT block obfuscation**
  - **Need to defend removal attack**

1) Combined with conventional logic locking techniques
2) Structural obfuscation
   - Add *n* MUX-based key-gates to increase interconnectivity
3) Functional obfuscation
   - *Add n* key-gates at the internal nets of Anti-SAT block
4) Re-synthesis the final design

Baseline Anti-SAT (BA)



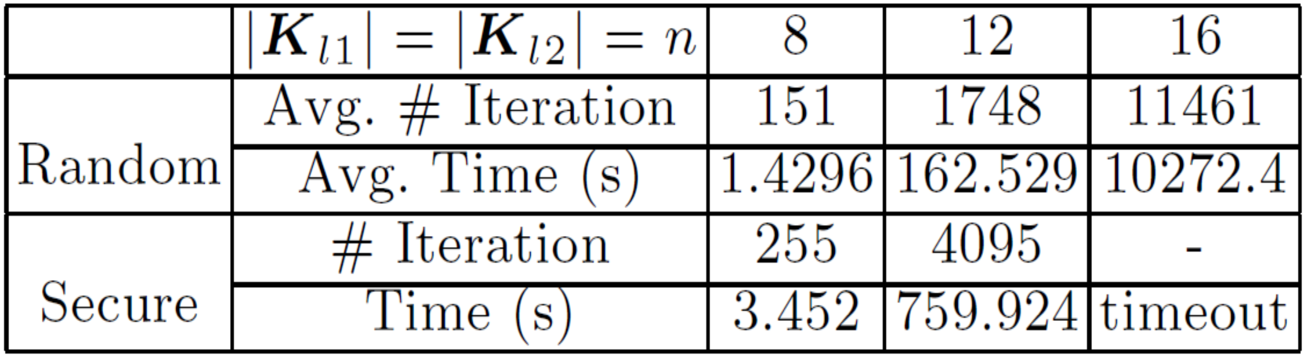
Obfuscated Anti-SAT (OA)



SAT Attack Results

- **Anti-SAT block location**

Location	Inputs	Output
Random	Randomly selected original wires	Another random wire that has a latter topological order
Secure	Primary Inputs	A random wire that has top 30% observability



	$\lvert\boldsymbol{K}_{l1}\rvert = \lvert\boldsymbol{K}_{l2}\rvert = n$	8	12	16
Random	Avg. # Iteration	151	1748	11461
	Avg. Time (s)	1.4296	162.529	10272.4
Secure	# Iteration	255	4095	-
	Time (s)	3.452	759.924	timeout

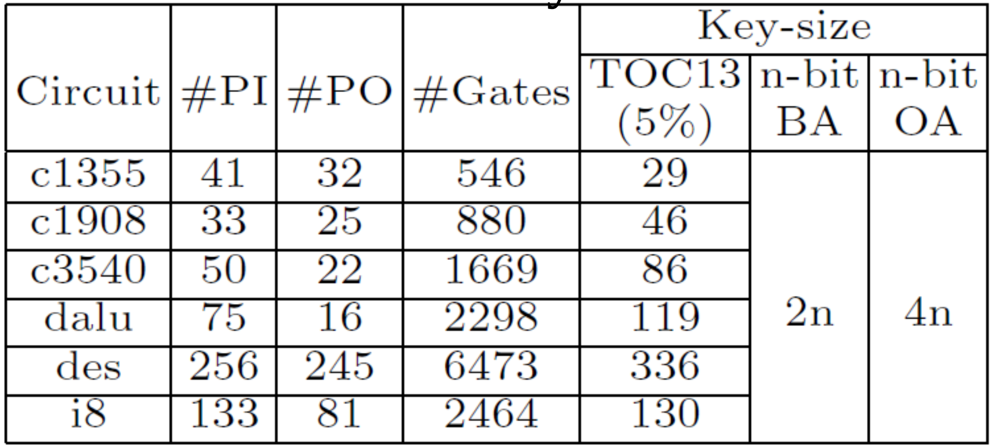
(10 hrs)

Secure location results in ~2X iterations and ~3X execution time

SAT Attack Results

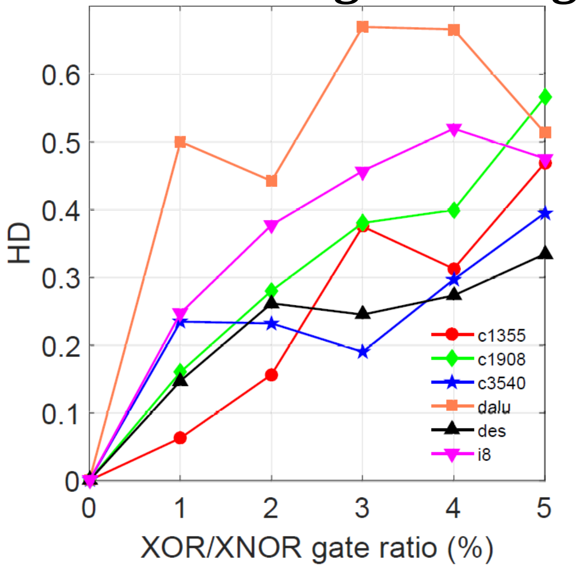
- **Anti-SAT block application**
  - 6 benchmarks for ISCAS85 and MCNC (500+ ~ 6000+ gates)
  - Three setups:
    - TOC13: insert XOR/XNOR gates at the original netlist
    - TOC13 (5%) + *n*-bit baseline Anti-SAT (*n*-bit BA)
    - TOC13 (5%) + *n*-bit obfuscated Anti-SAT (*n*-bit OA)

Benchmark and key-size information



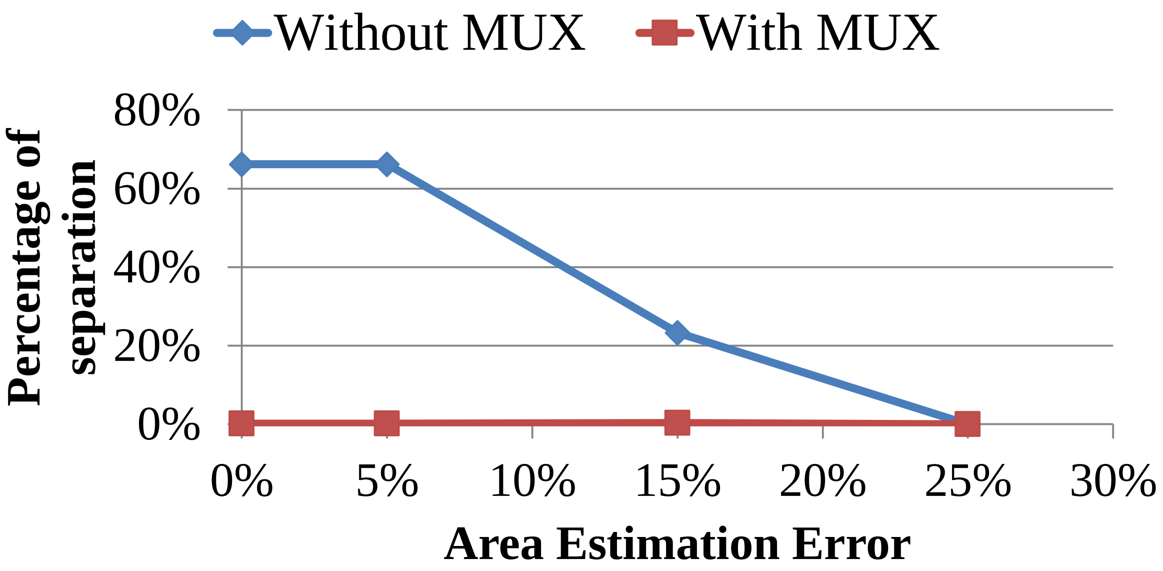
Circuit	#PI	#PO	#Gates	Key-size		
				TOC13 (5%)	n-bit BA	n-bit OA
c1355	41	32	546	29	2n	4n
c1908	33	25	880	46		
c3540	50	22	1669	86		
dalu	75	16	2298	119		
des	256	245	6473	336		
i8	133	81	2464	130		

TOC13 logic locking



Obfuscation Results

- **Anti-SAT block obfuscation**
  - Attack: use min-cut partitioning to isolate the Anti-SAT block*
  - Metric: percentage of gates the Anti-SAT block that are isolated and separated to the smaller partition
  - With/without MUX-based routing network
  - Area estimation error: 0% - 25%



* use a 14-bit Anti-SAT block