Deciphering SAT Attack on Computer Chips

NUST, College of Electrical and Mechanical Engineering

Department of Computer and Software Engineering

21st February, 2024

By: Dr. Muhammad Yasin

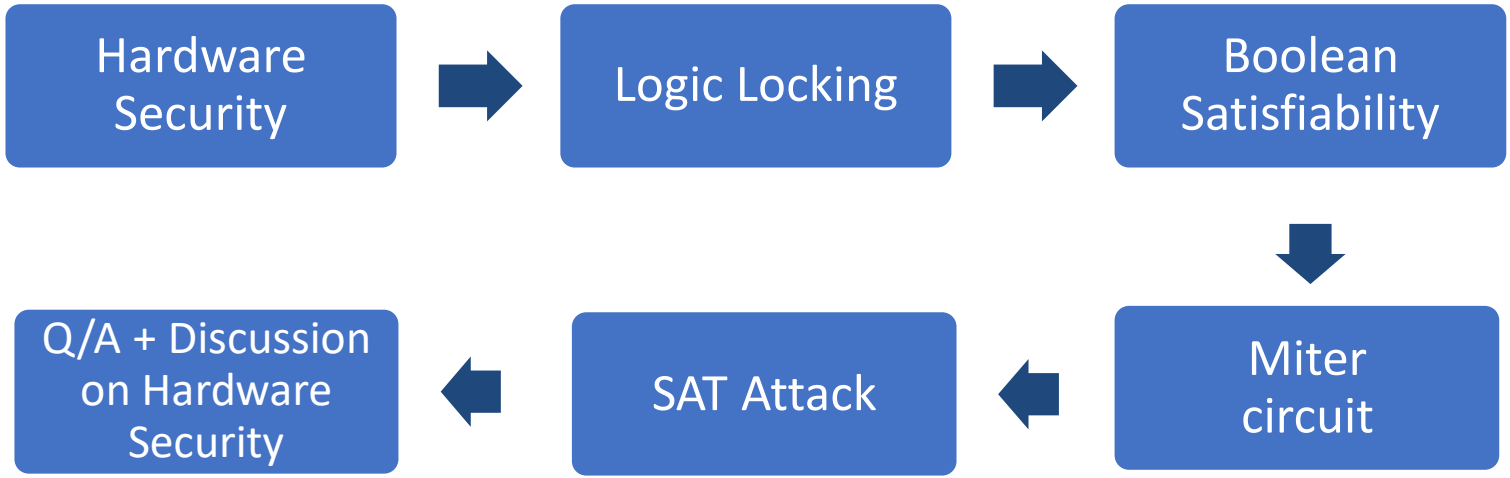
Organizer

Dr. Muhammad Yasin (Assistant Professor, NUST)

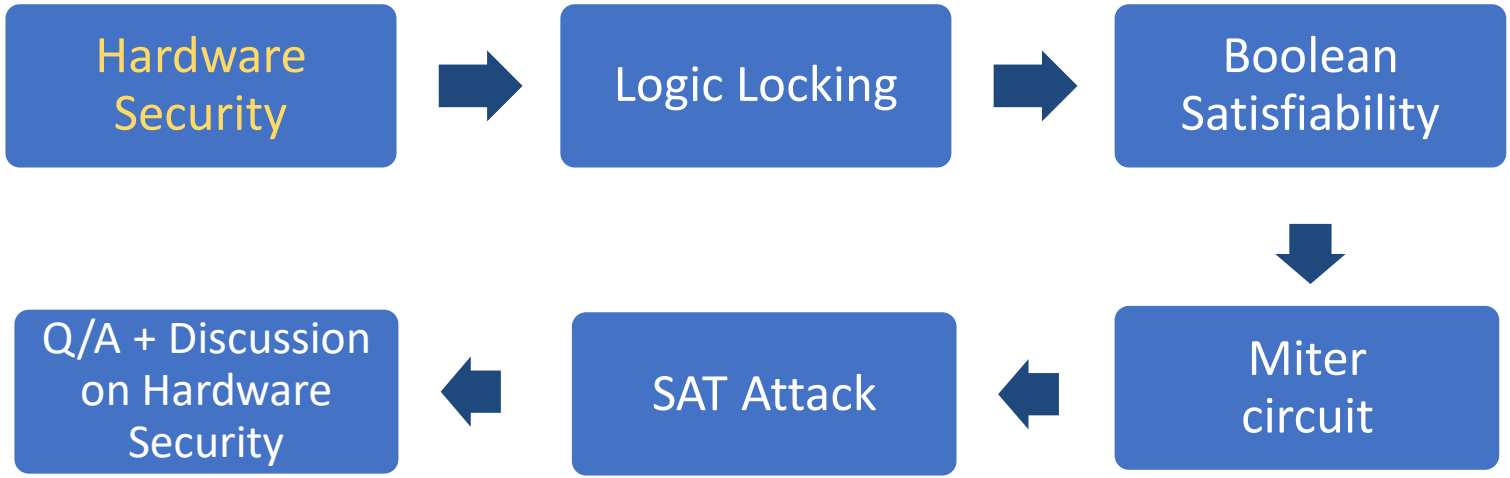
Co-organizers

Muhammad Mussa Kazim (Student of Computer Engineering, NUST)
Waseem Ghulam (Student of Computer Engineering, NUST)
Ameer Hamza (Student of Computer Engineering, NUST)

Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

Introduction to Hardware Security

1 **What is Hardware Security ?**

- Protection of electronic systems, circuits, and devices from threats
- Ensuring integrity, confidentiality, and availability.

2 **Importance of Hardware Security?**

- Prevents unauthorized access and breaches.
- Protects against hardware Trojans and side-channel attacks.
- Ensures trust in critical systems.

Key Figures in the Field of Hardware Security (US)

Mark Tehranipoor
University of Florida

Jim Plusquellic
University of New Mexico

Farinaz Koushanfar
University of California

Swarup Bhunia
University of Florida

Ramesh Karri
New York University

Yousef Iskander
Microsoft

Saverio Fazzari
Booz Allen Hamilton

Key Figures in the Field of Hardware Security (Europe/Asia)

Ozgur Sinanoglu
NYUAD

Michail Maniatakos
NYUAD

Lilas Alrahis
Khalifa University

Aijiao Cui
Harbin Institute of Technology

Debdeep Mukhopadhyay
IIT, Kharagpur

Key Figures in the Field of Hardware Security (Pakistan)

Osman Hasan
NUST

Ayesha Khalid
Queen's University, Belfast

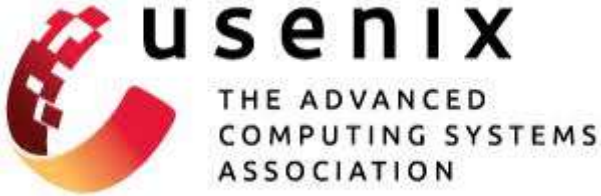
Zain ul Abideen
Carnegie Mellon University

Muhammad Shafique
NYUAD

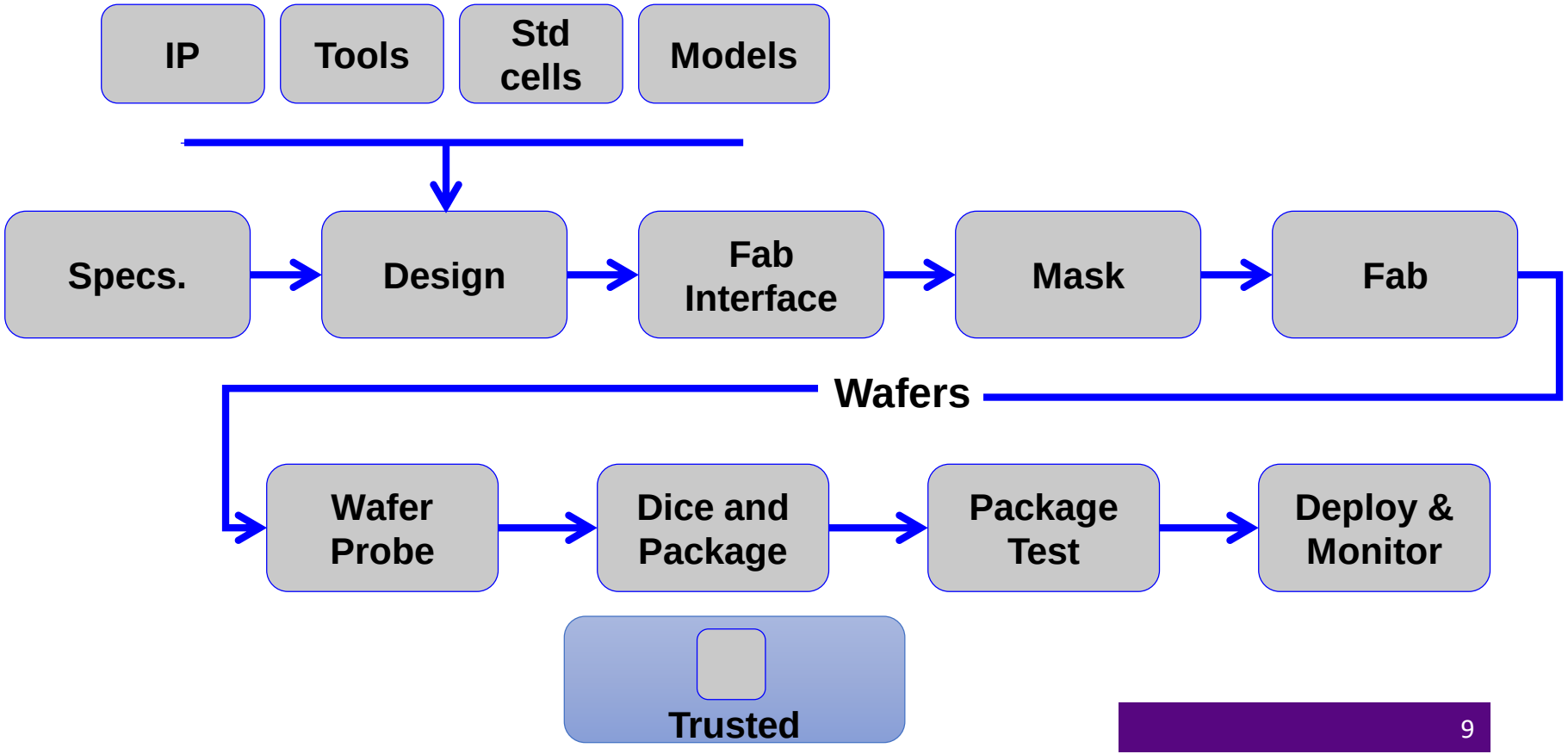
Fareena Saqib
UNC Charlotte

Haroon Waris
NECOP

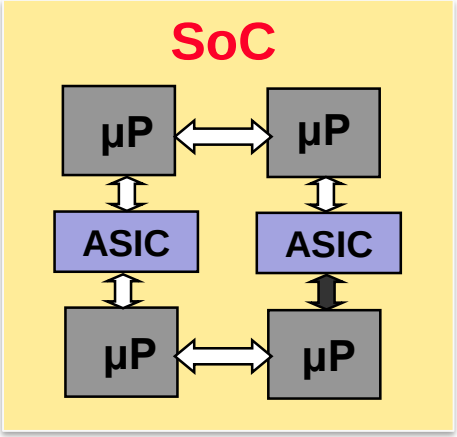
Some Famous Conferences Held on Hardware Security



Traditional IC Design and Process Flow



Globalized IC supply chain



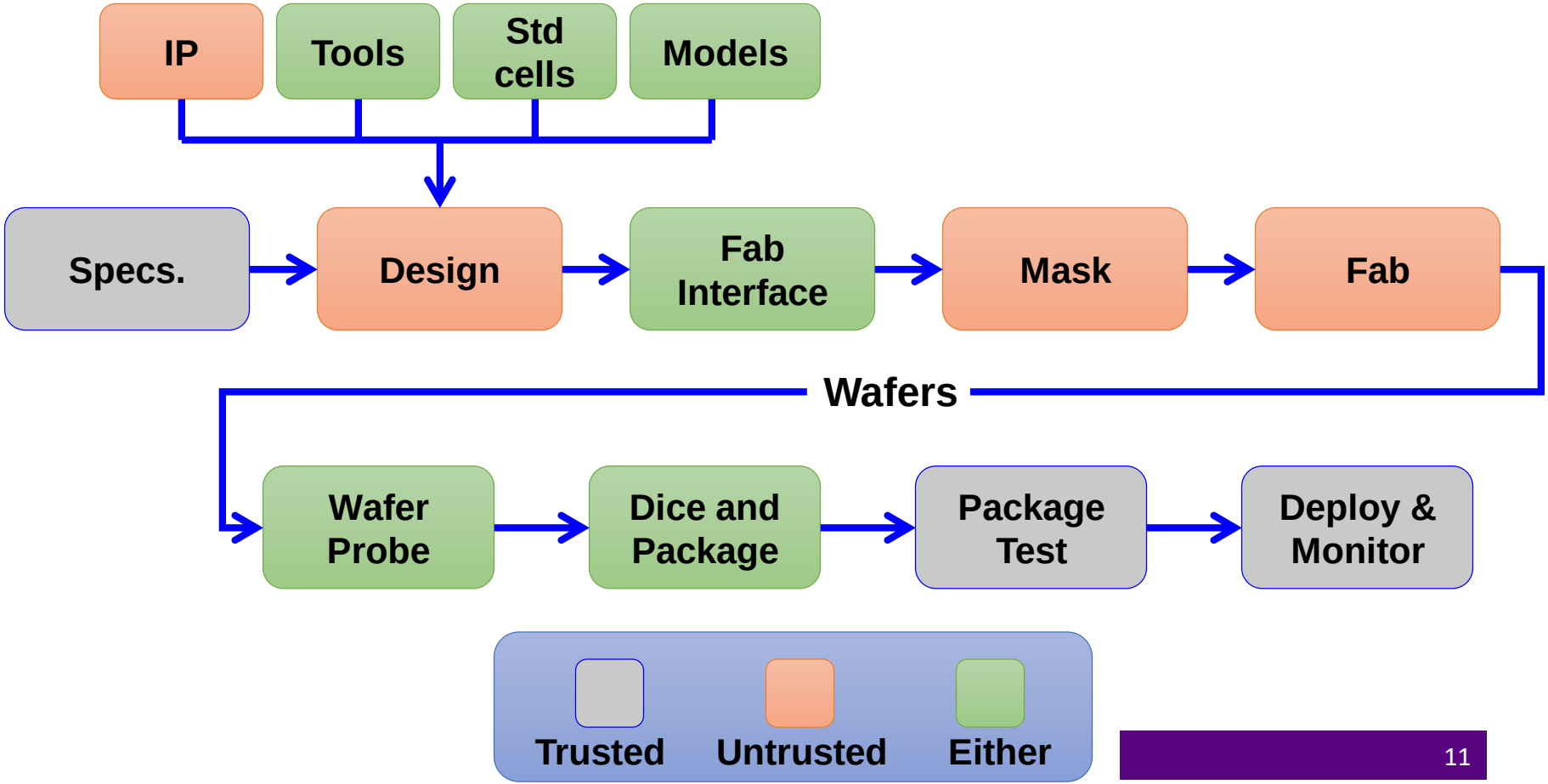
- Economic concerns
- Time-to-market
- Design complexity

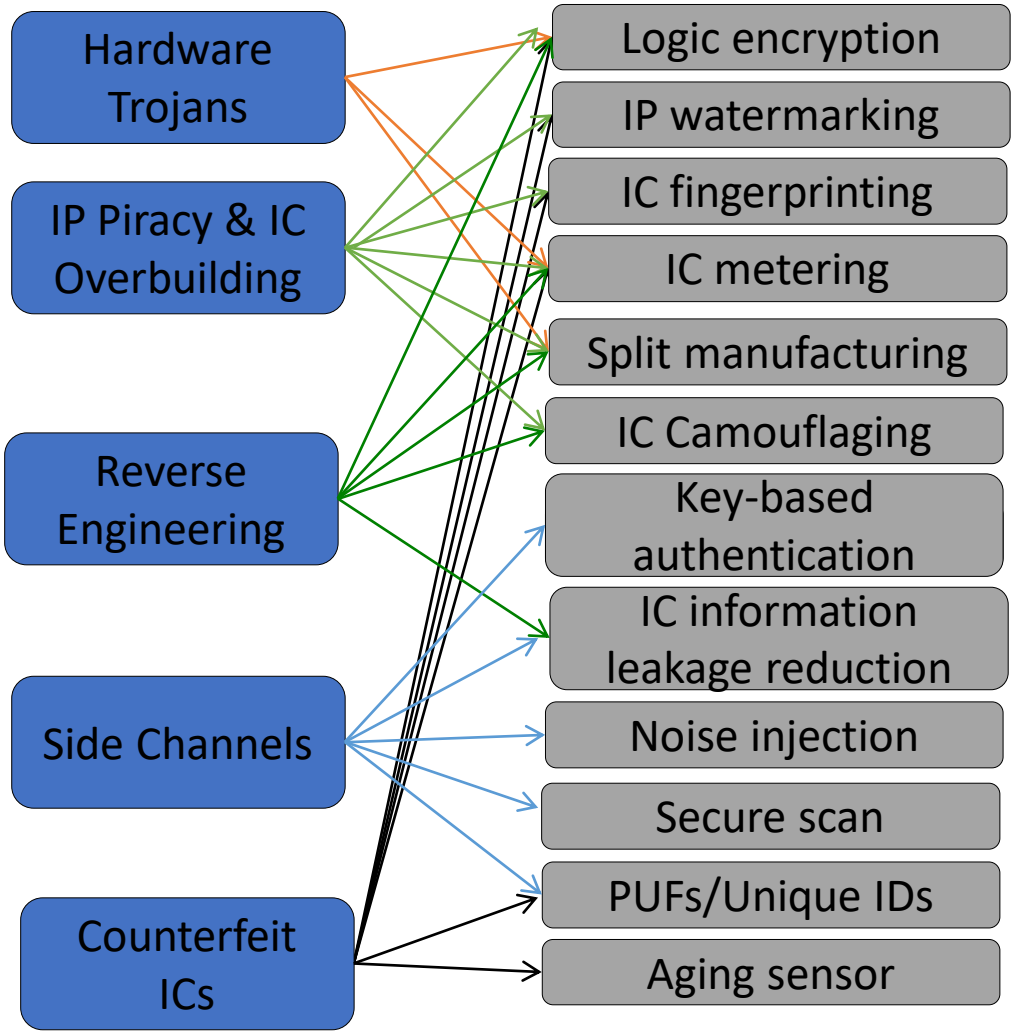
Design → Fabrication → Test/Assembly → Distribution

http://www.wallpapers-web.com/world-map-with-countries-wallpapers/5705193.html

Security vulnerabilities

Evolved IC Design and Process Flow





Hardware Trust Issues

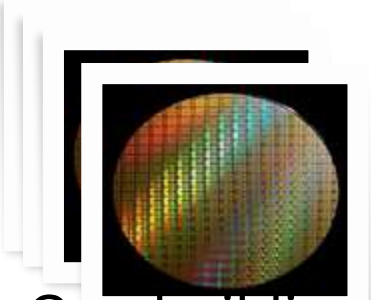
- Globalized IC supply Chain
- Active area of research
- Attacks and corresponding measures
- Focus of my work:
  - Logic Encryption

Threat landscape

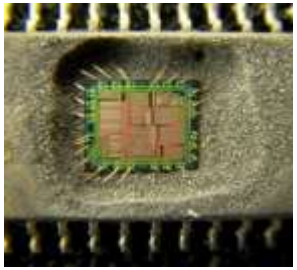
Real Fake

Counterfeiting

IP Piracy



Overbuilding



Reverse Engineering

Hardware Trojans

Threat landscape

Real Fake

Impact

- Loss of revenue ~ $4 billion annually
- Loss of trust
- Unreliable consumer electronics

Reverse Engineering

Hardware Trojans

IP Piracy and Reverse Engineering

Threat:

- **Access to a resource**
  - **e.g. netlist or layout**
- **Copy/use it illegally**

Defense:

- **Detect piracy and pursue legal action**
- **Prevent piracy by protecting /locking the design**

Logic encryption:

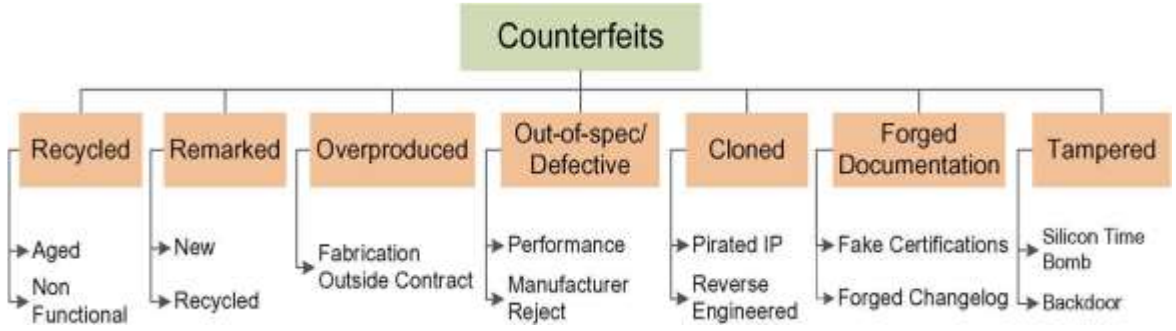
- Add extra logic to ‘lock’ the design
- Combinational logic encryption
  - Inserting additional gates (XOR/XNOR)
  - Replace gates with memories
- Sequential logic encryption

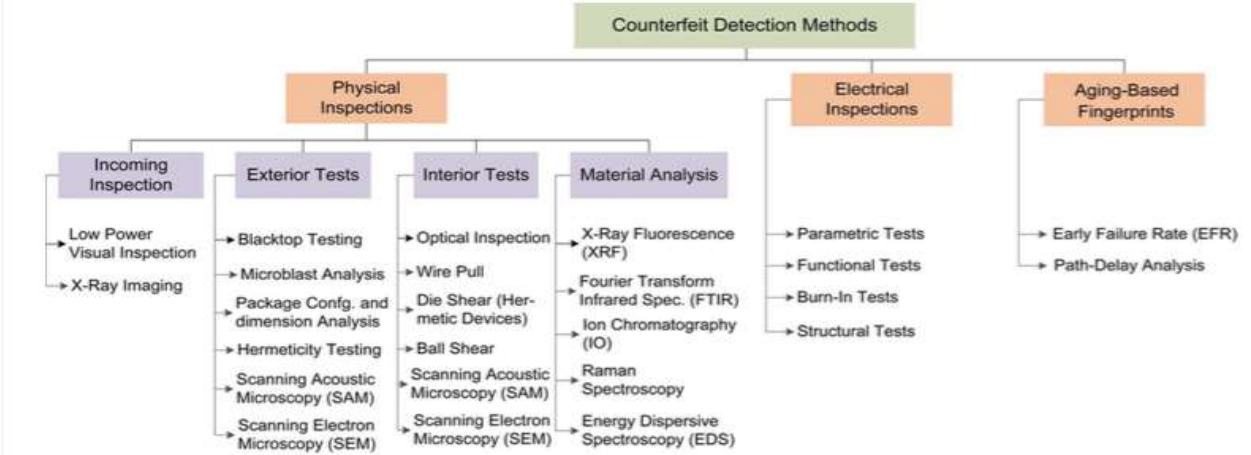
Primer on Hardware Security (Rostami et al.)

- **IP Piracy and Overbuilding**
  - Watermarking
    - Digital signatures
  - Fingerprinting
    - Reveals source of piracy
    - Use PUFs to generate IC signatures
  - Obfuscation
    - Hides functionality and implementation
    - Sequential / combinational
  - Metering
    - Post-fabrication tracking
  - Split Manufacturing
    - FEOL and BEOL in separate foundries
- **Hardware Trojans**
  - Invasive
    - Render IC unusable
  - Non-invasive
    - Side-channel analysis
      - Delay, dynamic leakage, thermal profiling
    - Logic Testing
- **Reverse Engineering**
  - IC Camouflaging
- **Side-Channel Attacks**
  - Leakage Reduction
  - Noise Injection
  - Key Update
- **Counterfeiting**
  - Hardware metering and Auditing
  - Device Aging sensors

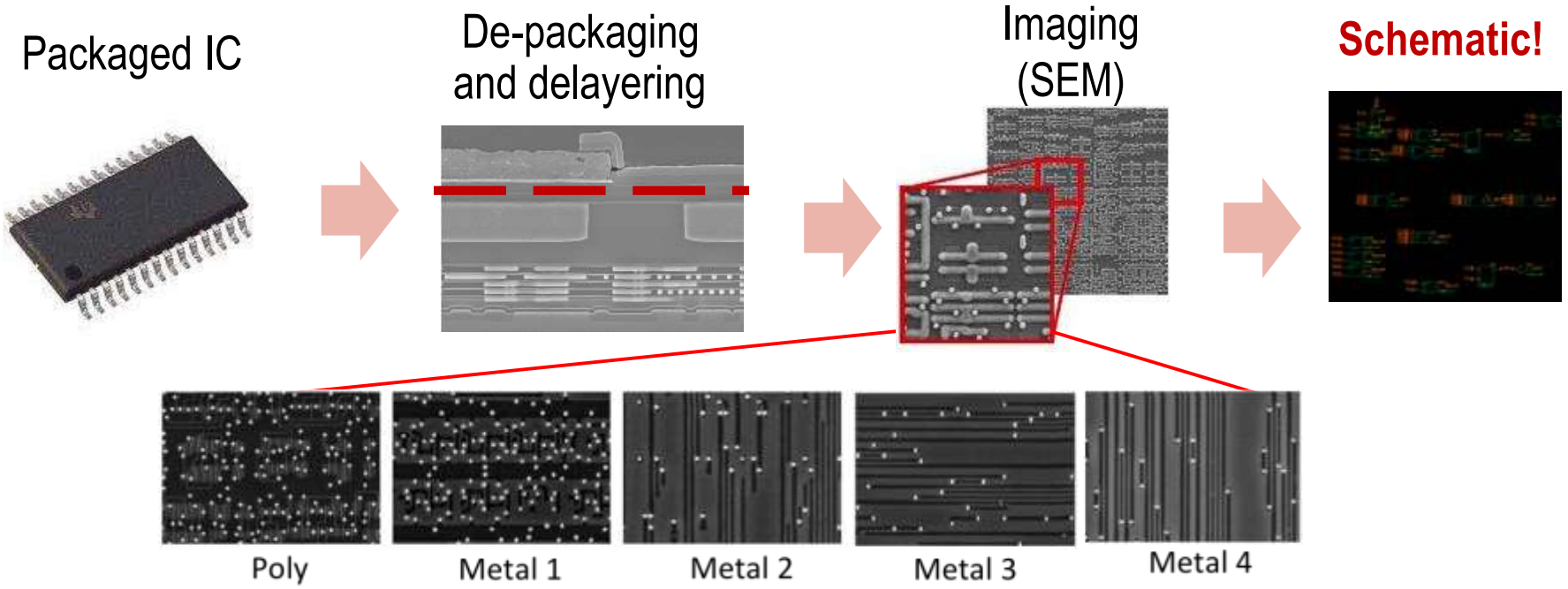
Counterfeit IC's: A Rising Threat (Guin et al.)

- **Motivation:**
  - Recycled/remarked/overproduced, cloned, out-of-spec
  - Reliability and security issues
  - For government, industry and consumers





IC reverse engineering



- Smart card analysis by Texplained
  - 5 layers (1500 images per layer)
  - Time to reverse engineer < 3 weeks

O. Thomas, Black Hat 2015

FPGA Security: Motivation and Features

Threats:

- Cloning/Overbuilding
  - Copy the program and sell it
  - Contractor builds additional components with permission
- Reverse Engineering
  - Reverse engineer the bitstream
- Tampering
  - Modify/disable parts of application
- Spoofing
  - Adversary replaces the bitstream with his own

Solutions:

- Bitstream encryption
  - Prevents cloning
  - Different vendors have used different encryption algorithms: 3DES, AES etc.
- Bitstream encryption (cont'd)
  - Vendor software can provide the programmer with 1) encrypted bitstream 2) key-insertion file
  - Bitstream can be loaded in either either encrypted or unencrypted form
- Bitstream Authentication
  - One way message authentication used to establish trust and assure that application (bitstream) has not been altered
  - Encryption and authentication can also be integrated. This also allows for authentication key to send along with bitstream

Microcontrollers as (In) Security Devices

Motivation:

- Microcontrollers are ubiquitous
- Low-end devices
- Subject to a variety of attacks

- **Side Channel Attacks**

uC's exhibit stronger side-channel leakage compared to ASICs or FPGAs

- Simple Power Analysis (SPA)
  - One trace or an average over a few traces considered
  - RSA: square vs. multiply operation vary in energy consumption
- Template Attacks
  - Replace manual inspection in SPA with pattern matching and machine learning techniques
- Differential Power Analysis (DPA)
  - Analyze multiple traces with varying input data

- **Code Extraction Attacks**
  - With uC's, most IP is in form of program code
  - Protected typically by use of lock bits
  - However, this protection can be bypassed using:
    - Micropobing
    - Power glitching
    - UV light exposure
    - Fault injection
- **Side Channel Code Extraction Attacks**
  - Classical classification task
  - Train the algorithm (PCA, LDA , SVM) on individual instructions
  - Then classify the newly recorded traces to identify the instructions in a program

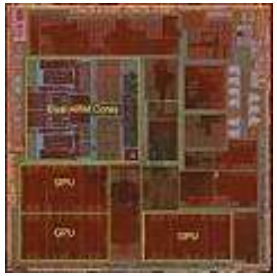
Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

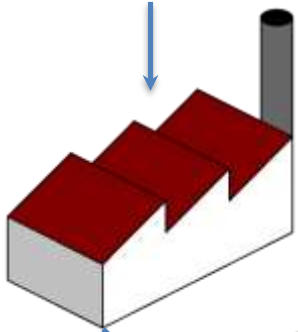
Traditional (insecure) IC design flow

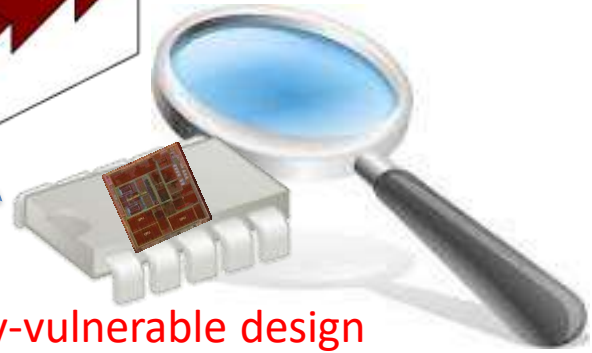
Design specifications

Design process



GDS-II files



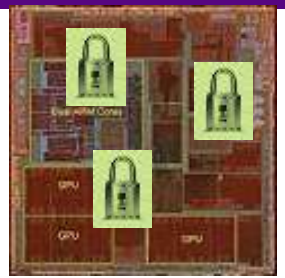


Piracy-vulnerable design

Tampered chips

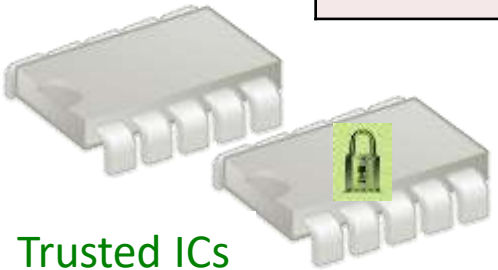
Secure IC design flow

Design specifications

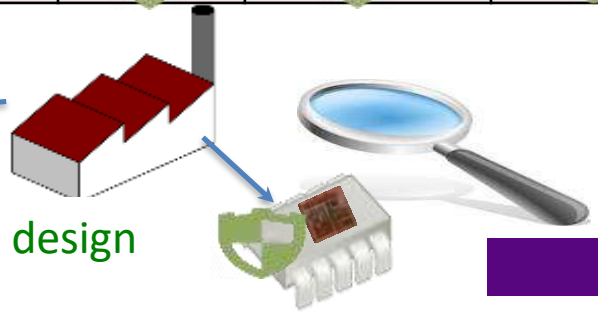


GDS-II files

Entity	Designer	Foundry	Test facility	End-user
Split manufacturing				
IC Camouflaging				
Logic locking				



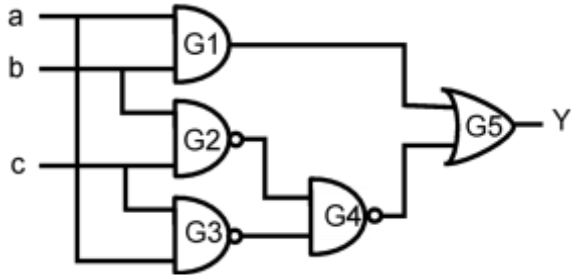
Trusted ICs



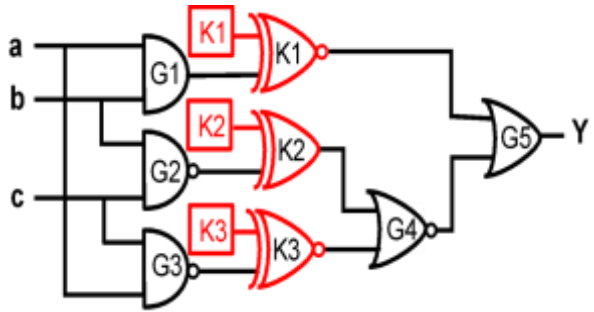
Piracy-shielded design

Logic locking

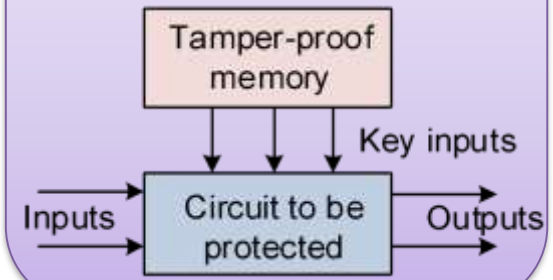
Original circuit



Locked circuit



Generic logic locked circuit

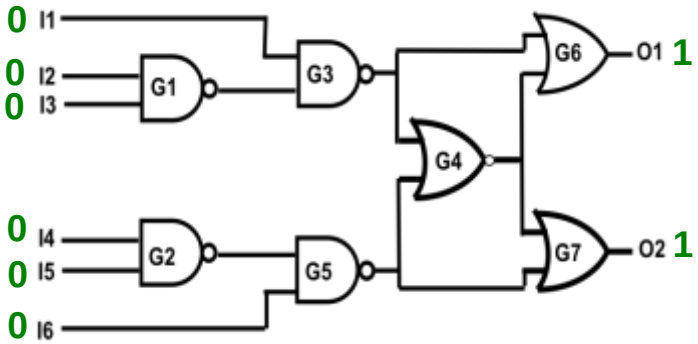


- Incorrect key ➔Incorrect output
- Thwarts reverse engineering-based piracy, overbuilding
  - Adversary does not know the secret (correct) key

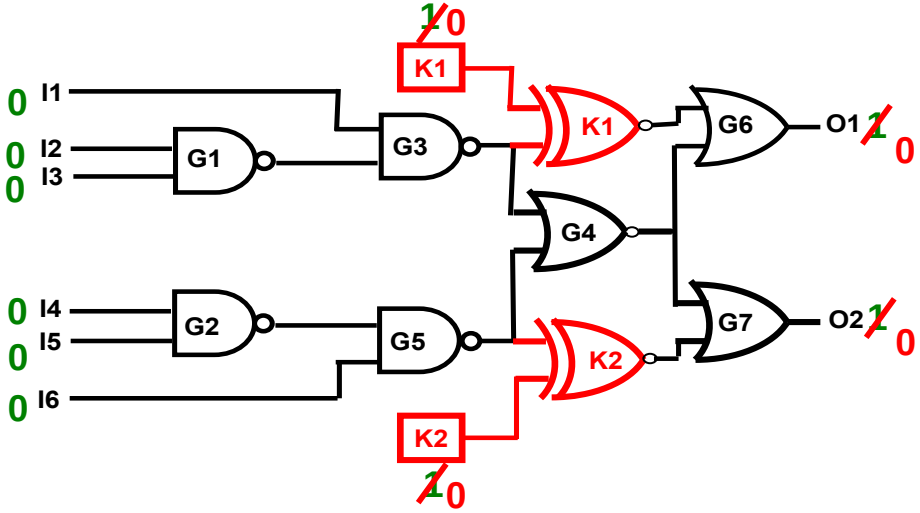
Protects the design (not the data)

Roy et al., DATE 2008. | Rajendran et al. , TCOMP 2015 | Rajendran et al. DAC 2012

Example of a locked circuit



Original circuit



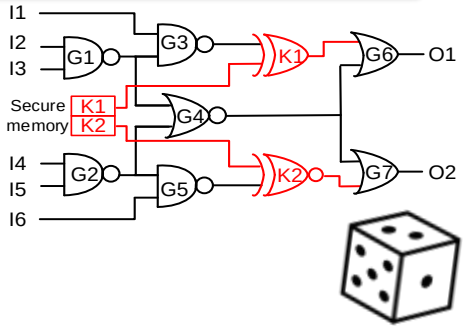
Locked circuit

“Older”Locking Techniques

Random LL (RLL)[1]

Key-gates at random locations in the netlist

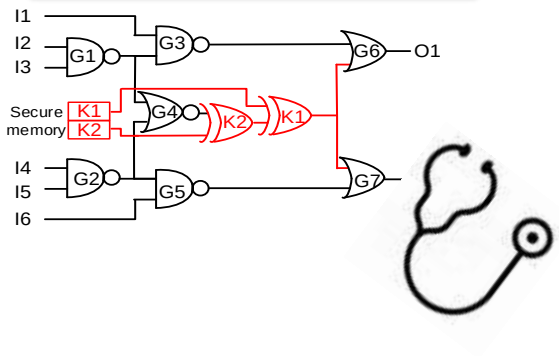
Key-gates uniformly distributed in the netlist



Fault analysis based LL (FLL)[2]

Key-gates at the most “influential” locations in the netlist

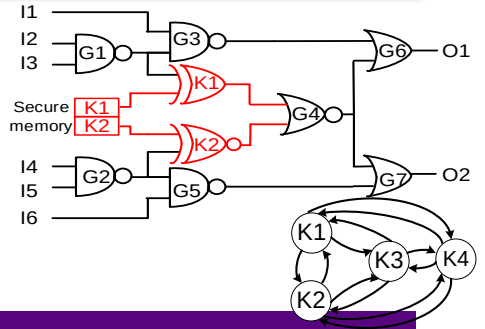
Key-gates tend to be localized and mostly back-to-back



Strong LL (SLL)[3]

Key-gates to hamper sensitization of individual key bits

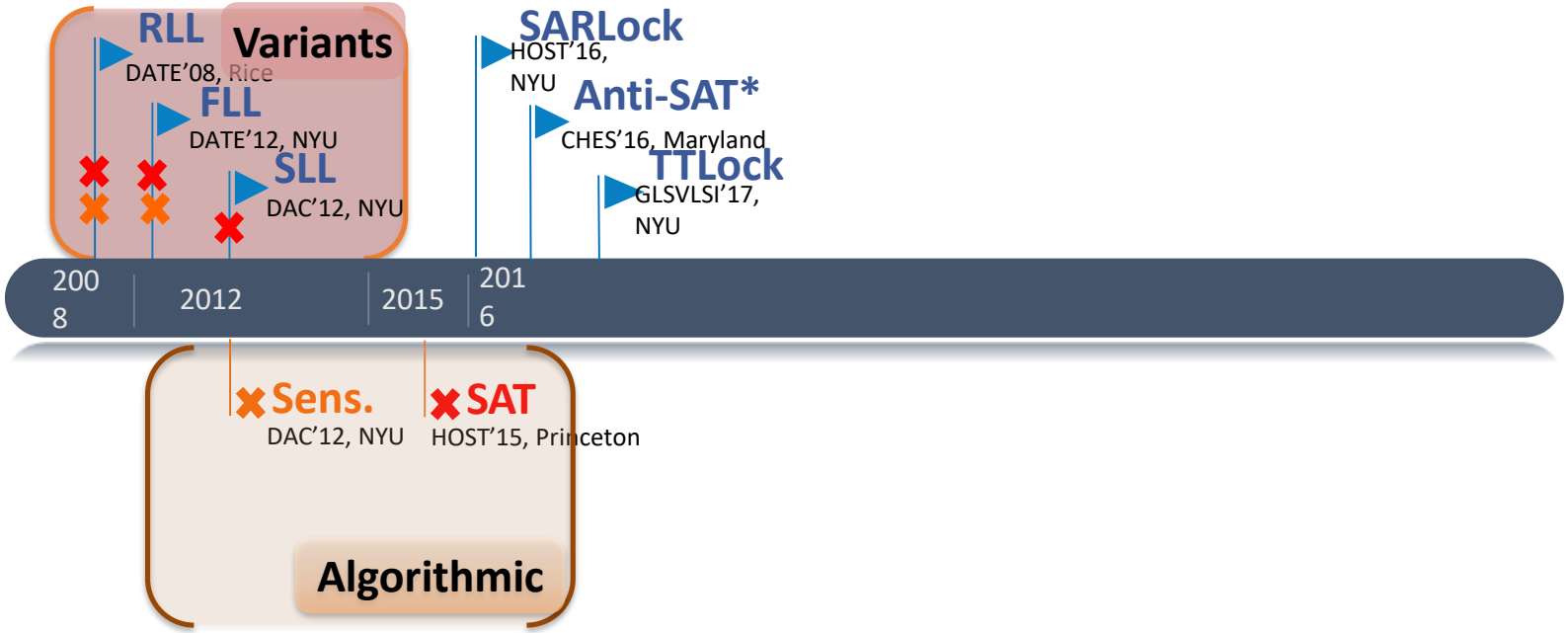
Key-gates localized in the netlist



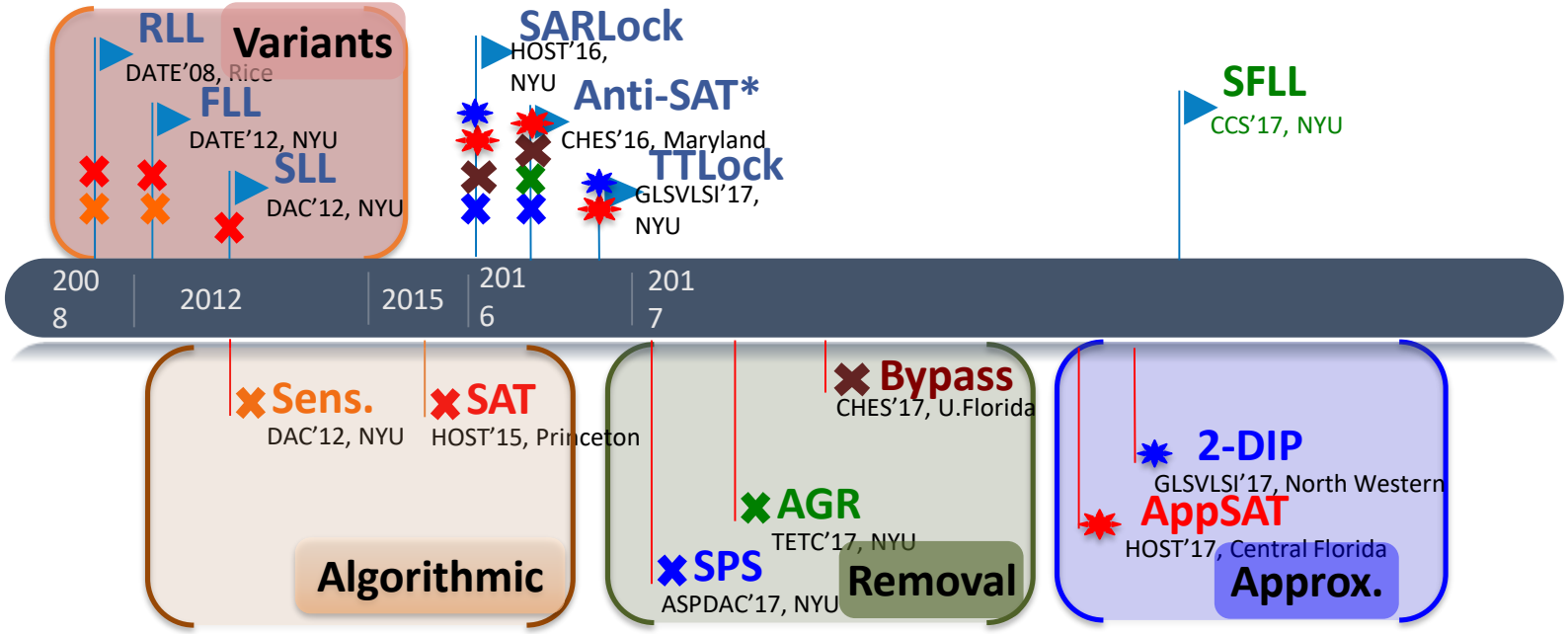
General Strategy of Attacks

Switch Presentation!

Evolution of logic locking (combinational)



Evolution of logic locking (combinational)



*- Each attack applies to different configurations of Anti-SAT

Research in Logic Locking

Attacks

Target specific locking techniques/algorithms

- Algorithmic
- Approximate
- Removal
- Side-channel

Defenses

Pre-SAT

- Random
- Fault-anal
- Strong

SAT attack

Post-SAT

- Point-function ba

Removal / approx. attacks

Overview of SAT Attack

Objective:

- Retrieve key for a locked circuit

Requirements:

- Locked circuit
- Oracle/functional IC

Strategy:

- Iteratively refine search space
- Use oracle output to filter out incorrect keys
- Extensive use of Boolean satisfiability solvers

About Inventor – Dr. Pramod Subramanyan

1984 - 2020

- Dr. Pramod Subramanyan was a researcher in **hardware security and formal methods**.
- He worked on SAT-based attacks while pursuing his Ph.D. at UC Santa Barbara
- He co-authored the **first paper on SAT attacks** in 2015.
- The paper, titled *"Evaluating the Security of Logic Encryption Algorithms"*, introduced **SAT-based decryption** of logic-locked circuits.

About Another Inventor – Dr. Siddharth Garg

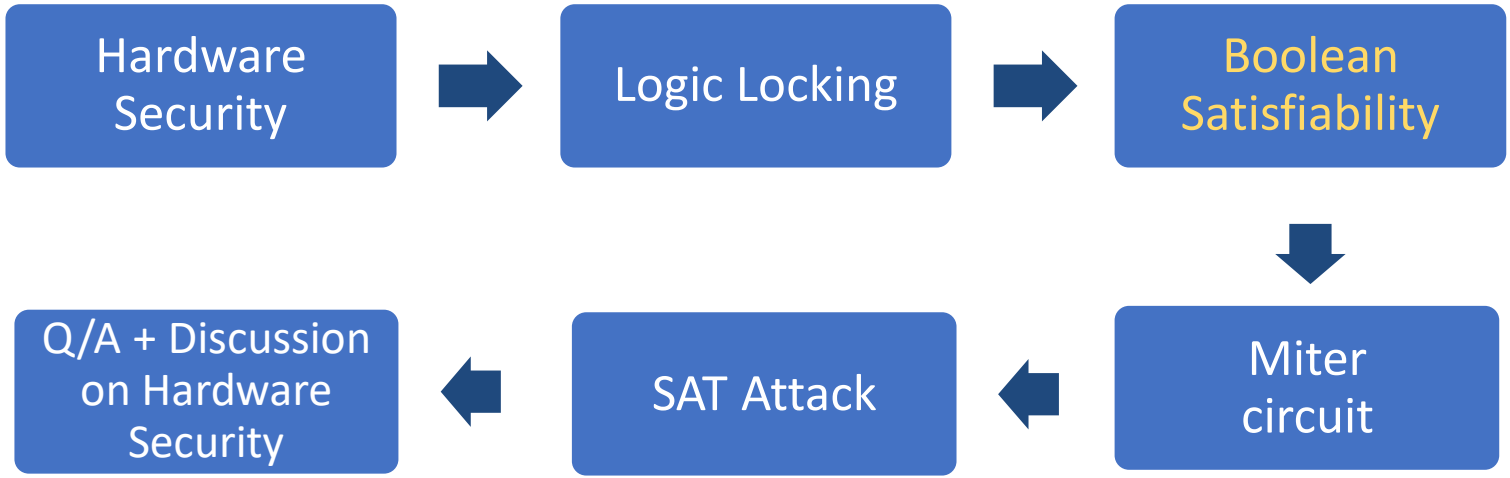
NYU Tandon Professor Siddharth Garg is "Brilliant"

POSTED:
SEPTEMBER 6, 2016

Cybersecurity Researcher Named to Popular Science Magazine's "Brilliant 10" for His Innovations That Protect Microchips

- Mohamed El Massad, Siddharth Garg, Mahesh V Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes", **NDSS 2015**

Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

Boolean Satisfiability (Used in SAT)

The Boolean Satisfiability Problem determines whether a given Boolean formula can evaluate to true for some assignment of variables. For example, the formula $(a \wedge b) \vee \neg c$ is satisfied by $(a,b,c)=(1,1,0)$.

Definition

SAT determines if a Boolean formula can be true with some variable assignments

CNF Requirement

SAT solvers need formulas in Conjunctive Normal Form (CNF), a conjunction of clauses.

1. Basic Structure of a DIMACS CNF File

A DIMACS CNF file has three parts:

- **Comments** (optional, start with `c`).
- **Problem line** (starts with `p cnf` followed by the number of variables and clauses).
- **Clauses** (lists of literals, ending with `0`).

2. Example 1: Simple Formula

Formula:

$$(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1)$$

DIMACS CNF File

```
c simple v3_c2.cnf
c
p cnf 3 2              # 3 variables , 2 clauses
1 -3  0                # Clause 1: X1 OR NOT X3
2  3 -1 0              # Clause 1: X2 OR X3 OR NOT X1
```

Explanation:

- **Variables**: $x_1 = 1, x_2 = 2, x_3 = 3$.
- **Negations**: `-3` means $\neg x_3$.
- **Clauses**: Each ends with `0`.
  - `1 -3 0` → $x_1 \vee \neg x_3$.
  - `2 3 -1 0` → $x_2 \vee x_3 \vee \neg x_1$.

3. Example 2: Larger Formula

Formula:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$

```
c example cnf
p cnf 3 3          # 3 variables , 3 clauses
1 2 0              # Clause 1: x1 OR  x2
-2 3 0             # Clause 2: NOT x2 OR x3
-1 -3 0            # Clause 3: NOT x1 OR NOT x3
```

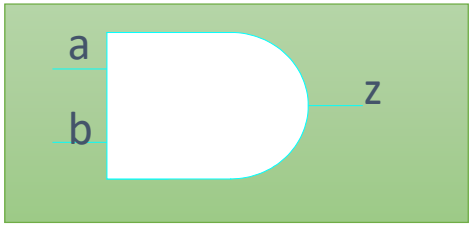
REAL WORLD EXAMPLE:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$$

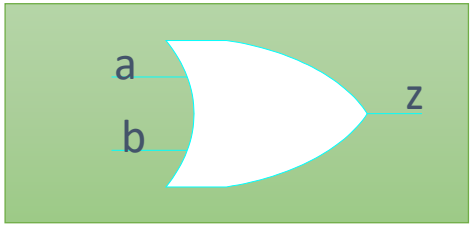
```
c example with 4 variables and 3 clauses
p cnf 4 3
1 -2 4 0
-1 3 0
-1 -3 0
2 -3 4 0
```

Tseitin Transformation

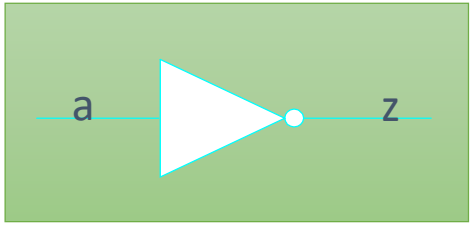
Tseitin Transformation converts a Boolean circuit into an equisatisfiable CNF formula by introducing auxiliary variables for gate outputs. This allows us to represent circuits in a format suitable for SAT solvers



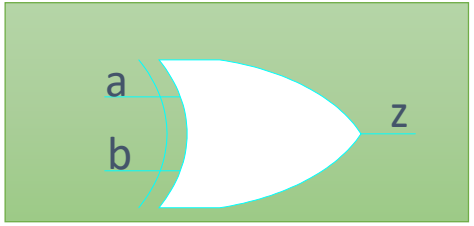
$\varphi = (a + z')\ (b + z')\ (a' + b' + z)$



$\varphi = (a' + z)\ (b' + z) \cdot (a + b + \neg z)$

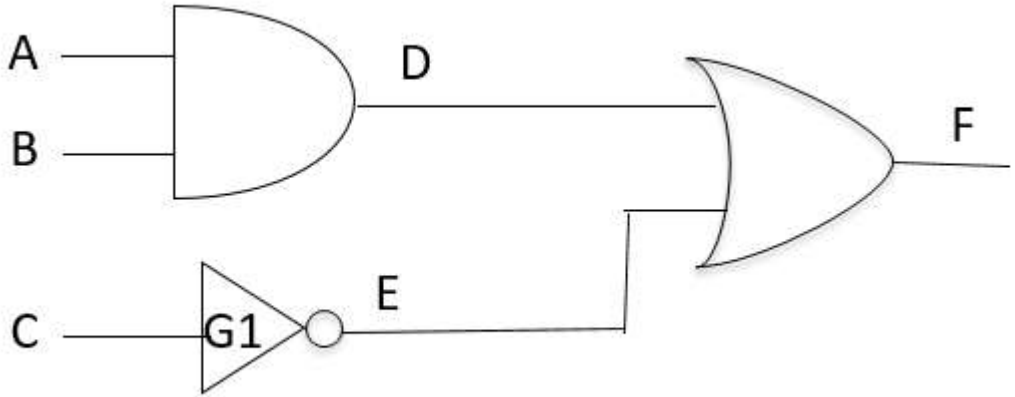


$\varphi = (a + z)\ (a' + z')$



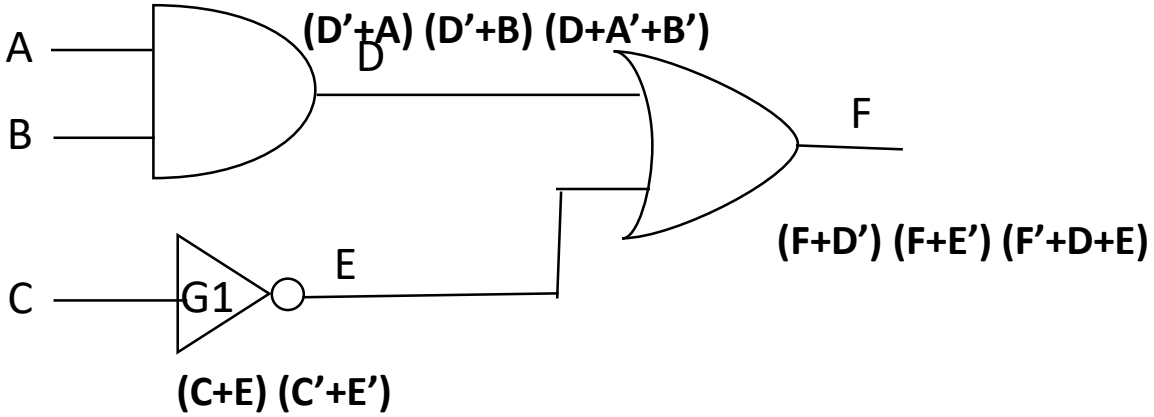
$\varphi = (a + b + \neg z)\ (a + b' + z)$
$(a' + b' + z')\ (a' + b + z)$

Practice CNF Question



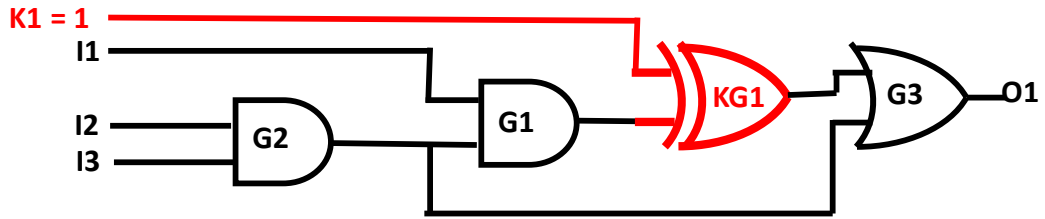
Solution of Practice CNF Question

- **Tseitin transformation:**
  - Write CNF for individual gates.

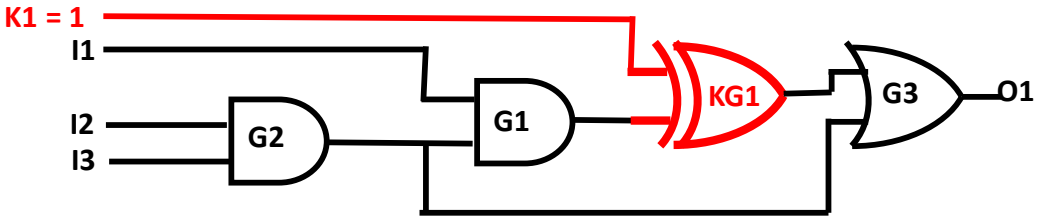


ϕ = (D'+A) (D'+B) (D+A'+B') (C+E) (C'+E') (F+D') (F+E') (F'+D+E)

Test CNF Circuit



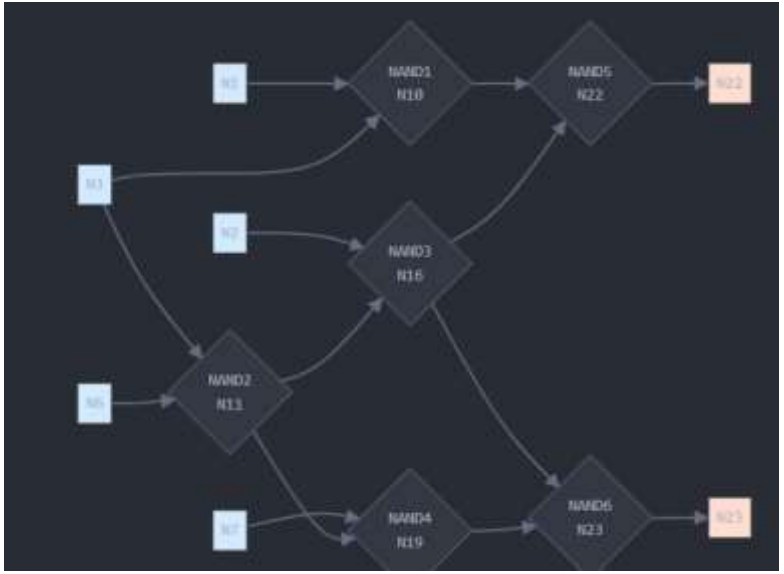
Solution of Test CNF Circuit



(K1) (G2'+I2) (G2'+I3) (G2+I2'+I3')
(G1'+I1) (G1'+G2) (G1+I1'+G2')
(O1+KG1') (O1+G2') (O1'+KG1+G2)
(KG1'+K1+G1) (KG1'+K1'+G1') (KG1+K1+G1') (KG1+K1'+G1)

C17 Circuit

- **Inputs**: $N1, N2, N3, N6, N7$
- **Outputs**: $N22, N23$
- **Internal Wires**: $N10, N11, N16, N19$
- **NAND Gates**:
  1. $N10 = \text{NAND}(N1, N3)$
  2. $N11 = \text{NAND}(N3, N6)$
  3. $N16 = \text{NAND}(N2, N11)$
  4. $N19 = \text{NAND}(N11, N7)$
  5. $N22 = \text{NAND}(N10, N16)$
  6. $N23 = \text{NAND}(N16, N19)$



Converting a NAND Gate to CNF

A NAND gate $z = \text{NAND}(a, b)$ is equivalent to:

$$z \leftrightarrow \neg(a \wedge b)$$

This can be written in CNF as:

$$(\neg z \vee \neg a \vee \neg b) \wedge (a \vee z) \wedge (b \vee z)$$

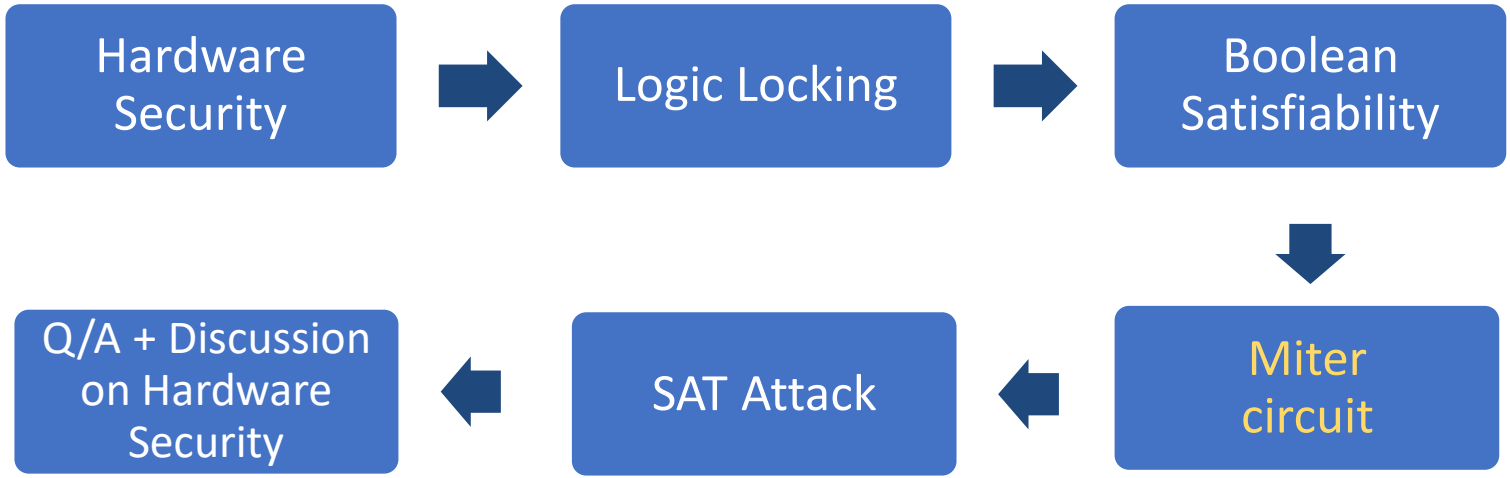
CNF Clauses for Each NAND Gate:

1. **NAND2_1**: $N10 = \text{NAND}(N1, N3)$
   - $(\neg N10 \vee \neg N1 \vee \neg N3) \wedge (N1 \vee N10) \wedge (N3 \vee N10)$
2. **NAND2_2**: $N11 = \text{NAND}(N3, N6)$
   - $(\neg N11 \vee \neg N3 \vee \neg N6) \wedge (N3 \vee N11) \wedge (N6 \vee N11)$
3. **NAND2_3**: $N16 = \text{NAND}(N2, N11)$
   - $(\neg N16 \vee \neg N2 \vee \neg N11) \wedge (N2 \vee N16) \wedge (N11 \vee N16)$
4. **NAND2_4**: $N19 = \text{NAND}(N11, N7)$
   - $(\neg N19 \vee \neg N11 \vee \neg N7) \wedge (N11 \vee N19) \wedge (N7 \vee N19)$
5. **NAND2_5**: $N22 = \text{NAND}(N10, N16)$
   - $(\neg N22 \vee \neg N10 \vee \neg N16) \wedge (N10 \vee N22) \wedge (N16 \vee N22)$
6. **NAND2_6**: $N23 = \text{NAND}(N16, N19)$
   - $(\neg N23 \vee \neg N16 \vee \neg N19) \wedge (N16 \vee N23) \wedge (N19 \vee N23)$

DIMACS CNF File

```
c c17 circuit CNF
p cnf 11 18
-6 -1 -3 0
1 6 0
3 6 0
-7 -3 -4 0
3 7 0
4 7 0
-8 -2 -7 0
2 8 0
7 8 0
-9 -7 -5 0
7 9 0
5 9 0
-10 -6 -8 0
6 10 0
8 10 0
-11 -8 -9 0
8 11 0
9 11 0
```

Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

What is a Miter Circuit ?

1 **Definition:**

Compares two circuits by XOR-ing their outputs

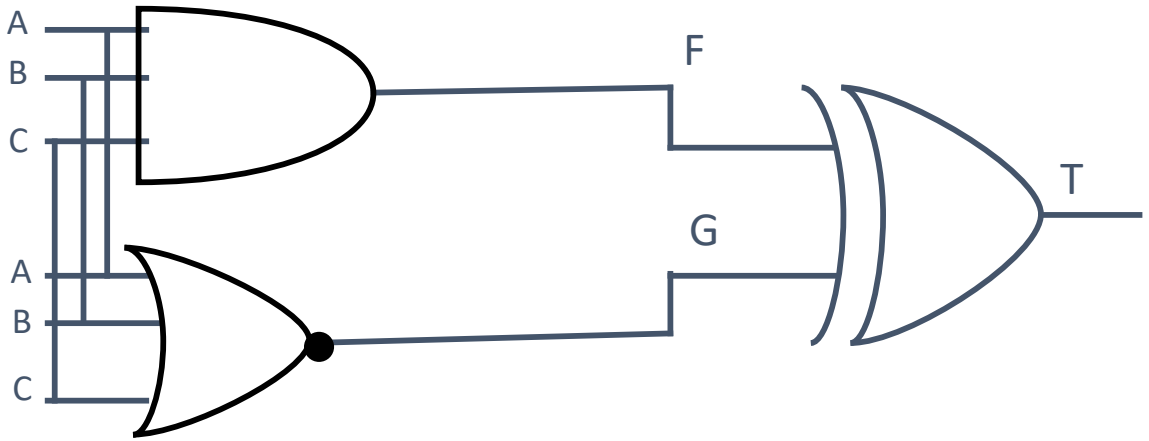
2 **Purpose:**

Checks if two circuits are identical.

3 **Output:**

z = 0 (identical), z = 1 (differ).

Miter Circuit Practice



Miter Circuit Practice

Solution:

a	b	c	F	G	T
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1

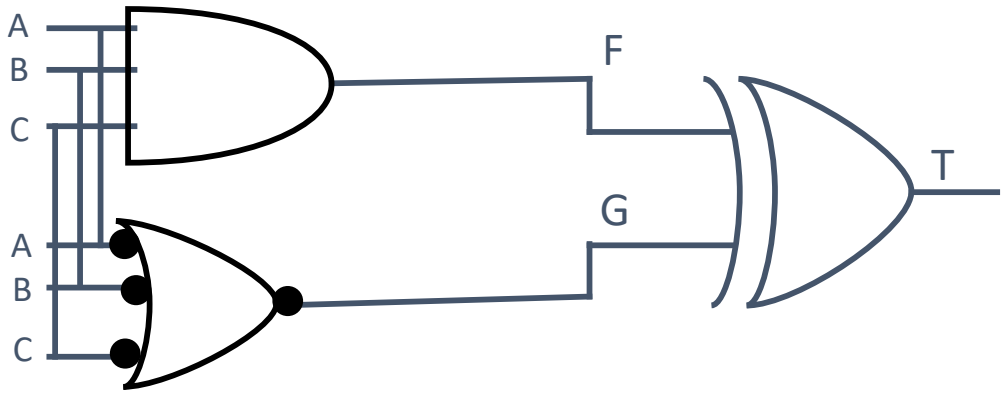
Miter Circuit Test

Circuit A

$y1 = a \wedge b \wedge c$

Circuit B

$y2 = \neg(\neg a + \neg b + \neg c)$



Miter Output

$z = y1 \oplus y2$

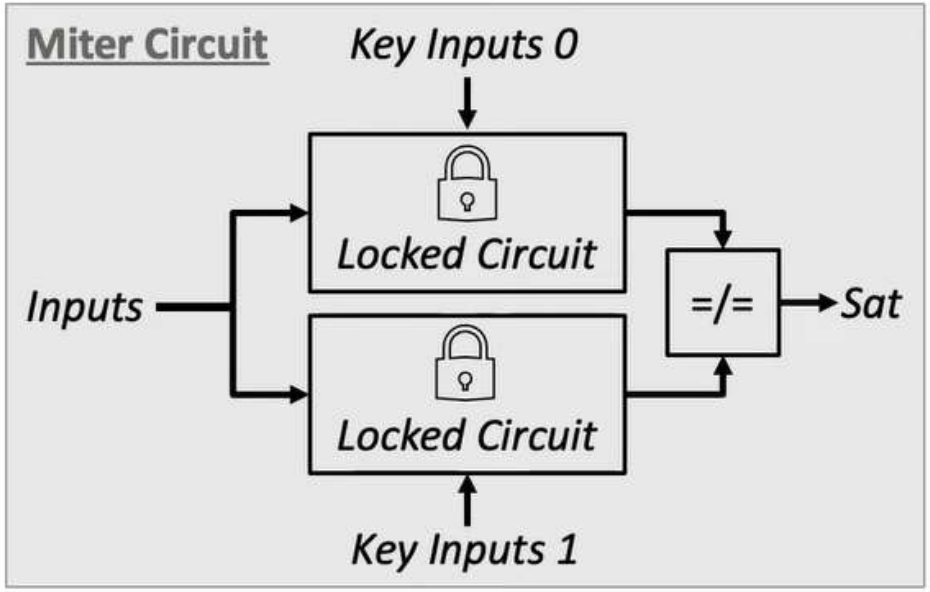
Miter Circuit Test Solution

Solution:

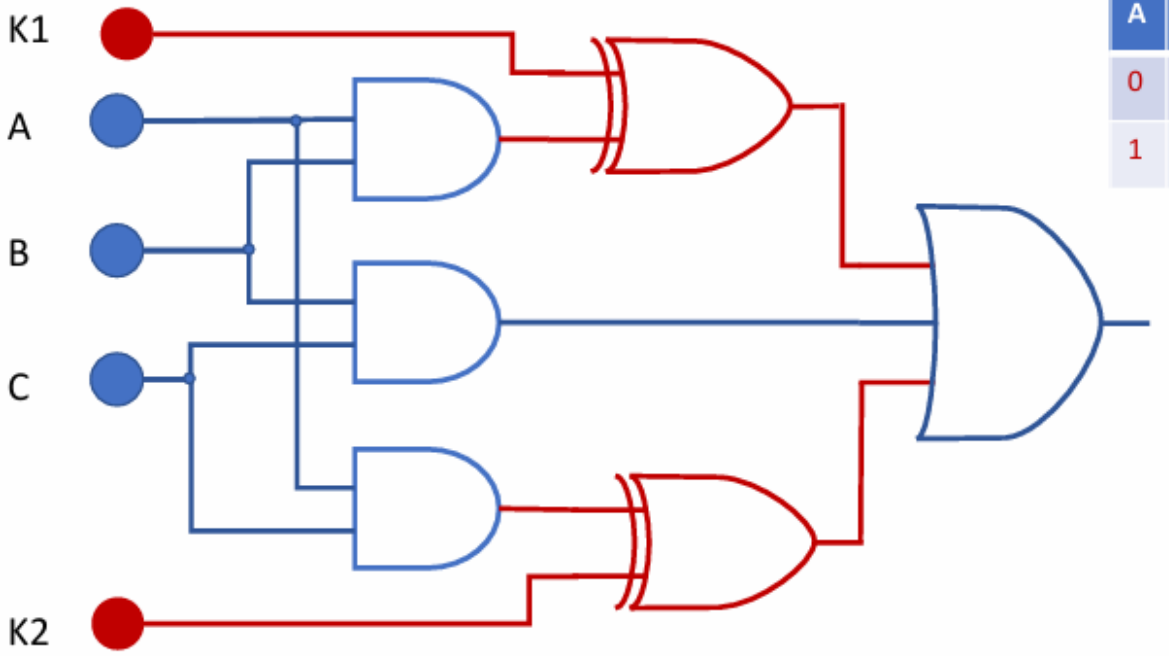
a	b	c	y1 = a∧b∧c	y2 =¬(¬a+¬b+¬c)	z= y1 ⊕ y2
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	0

Miter + SAT Solver

- ***Miter circuit*** + ***SAT solver*** → find inputs that ***distinguish sets of key-input values***



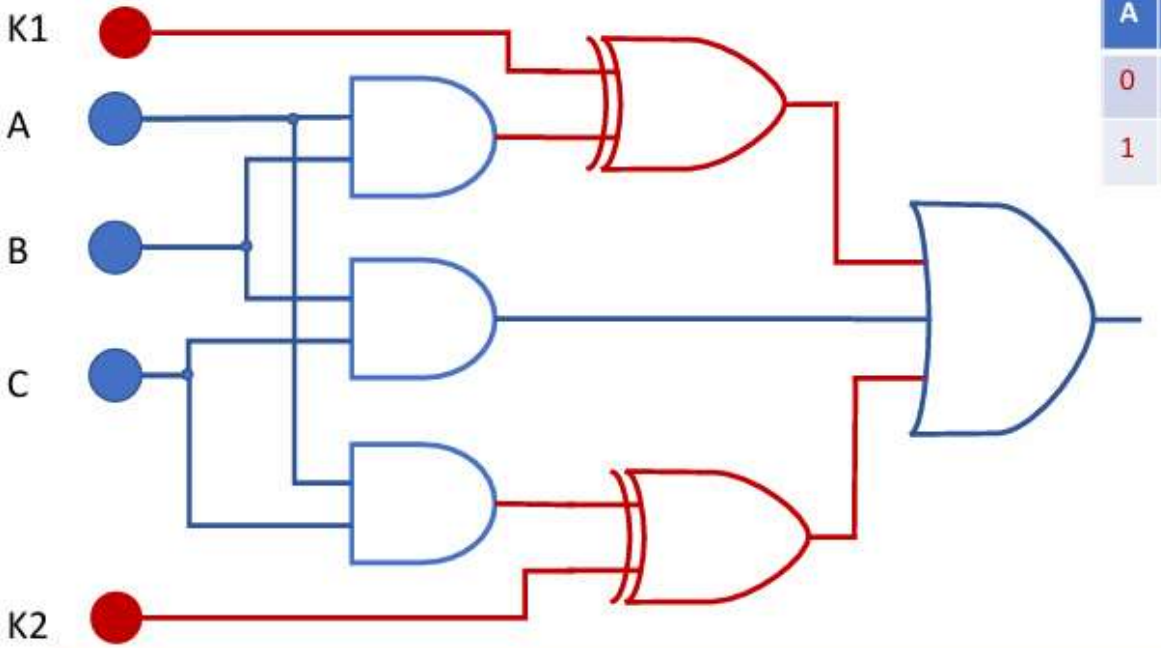
Key Idea in SAT attack: Distinguishing Inputs



A	B	C	Y
0	1	1	1
1	1	1	1

The above input vectors are useless because they result in the same output value for all keys

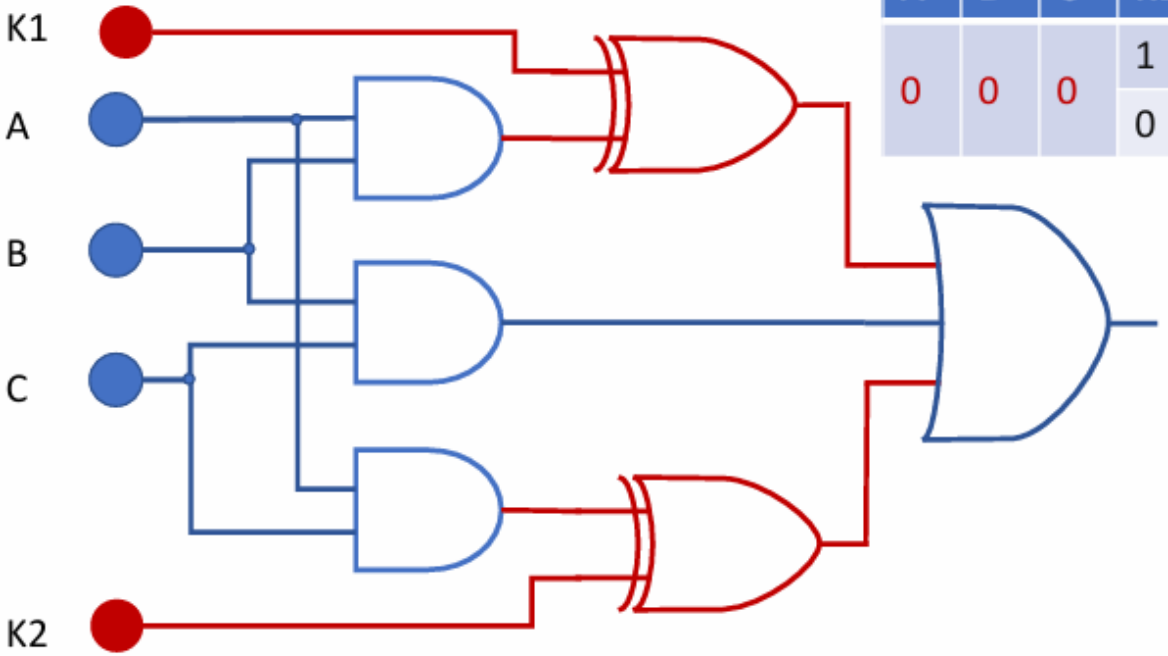
Key Idea in SAT attack: Distinguishing Inputs



A	B	C	Y
0	1	1	1
1	1	1	1

Want input vectors that can **distinguish** between at least two keys with **different behaviors**

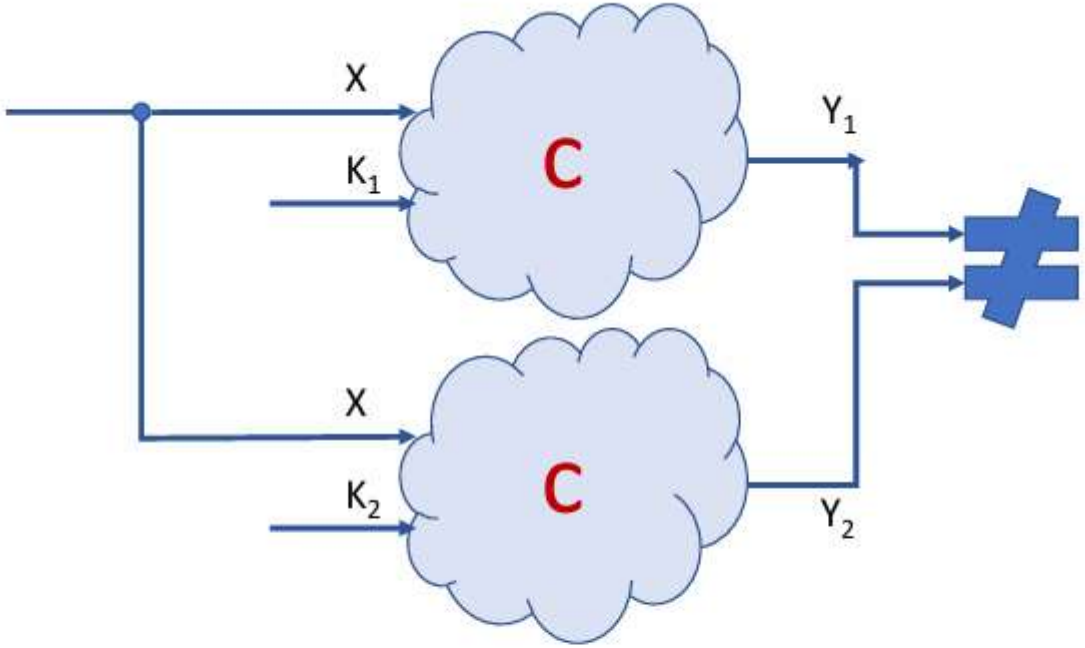
Key Idea in SAT attack: Distinguishing Inputs



A	B	C	K1	K2	Y
0	0	0	1	1	1
			0	0	0

Distinguishing input for two key values: an input vector that produces different outputs for these keys

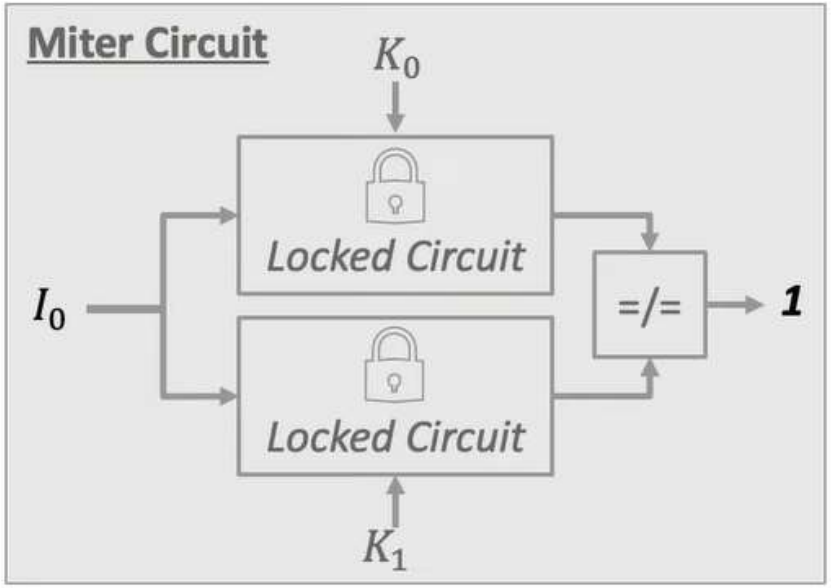
How do we compute Distinguishing Inputs

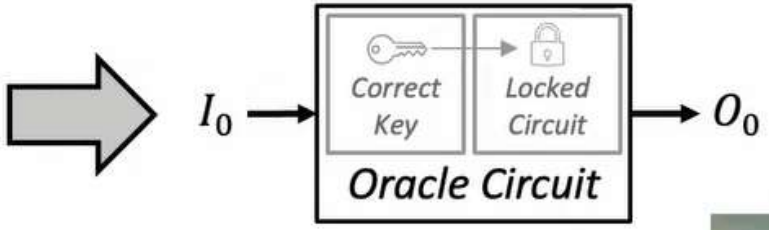


$$C(X, K_1, Y_1) \wedge C(X, K_2, Y_2) \wedge Y_1 \neq Y_2$$

Miter + SAT Solver

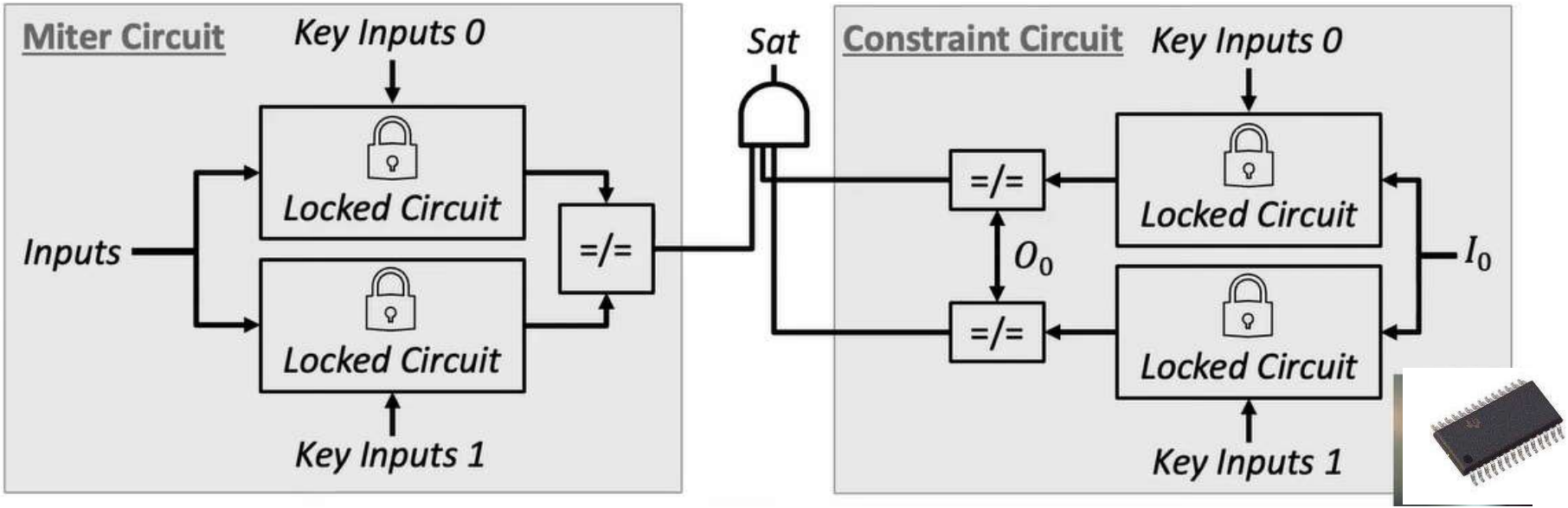
- ***Miter circuit*** + ***SAT solver*** → find inputs that ***distinguish sets of key-input values***
- Compare output with ***oracle*** to ***rule out invalid sets***





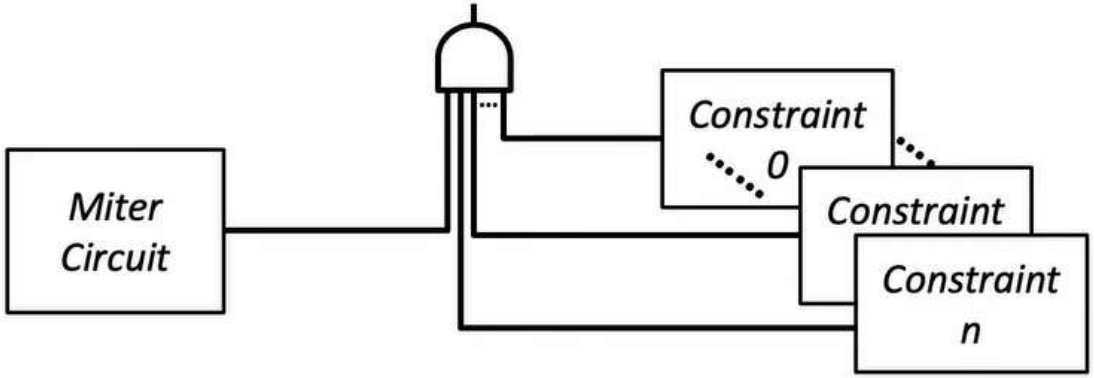
Miter + SAT Solver

- ***Miter circuit*** + ***SAT solver*** → find inputs that ***distinguish sets of key-input values***
- Compare output with ***oracle*** to ***rule out invalid sets***
- ***Add*** $\{I_0, O_0\}$ ***constraint*** to circuit model, ***repeat***



Miter + SAT Solver

- Each iteration new constraints are added
- Size of constraint increases with size of circuit
- Model can quickly become too large

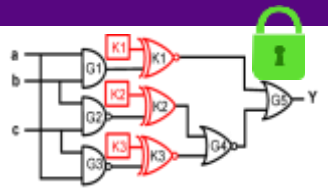


Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

SAT attack

Goal: Determine the secret key used for logic encryption



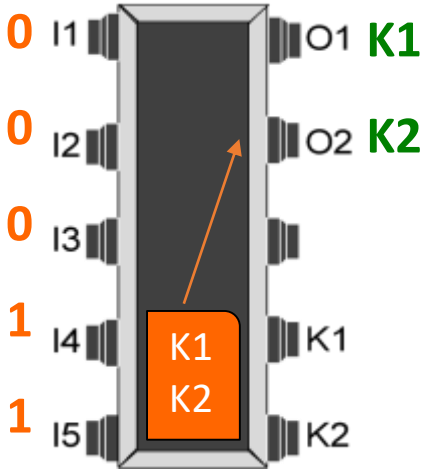
Functional IC

Attacker has:

- Locked Netlist
- Functional IC (with embedded key)

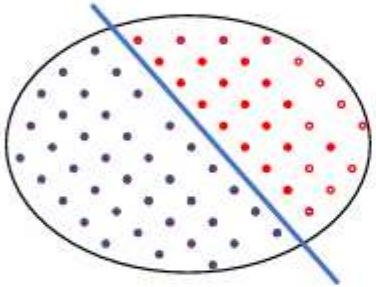
Attacker Does:

- Compute the attack patterns from the locked netlist
- Applies them on IC
- Infers key from responses

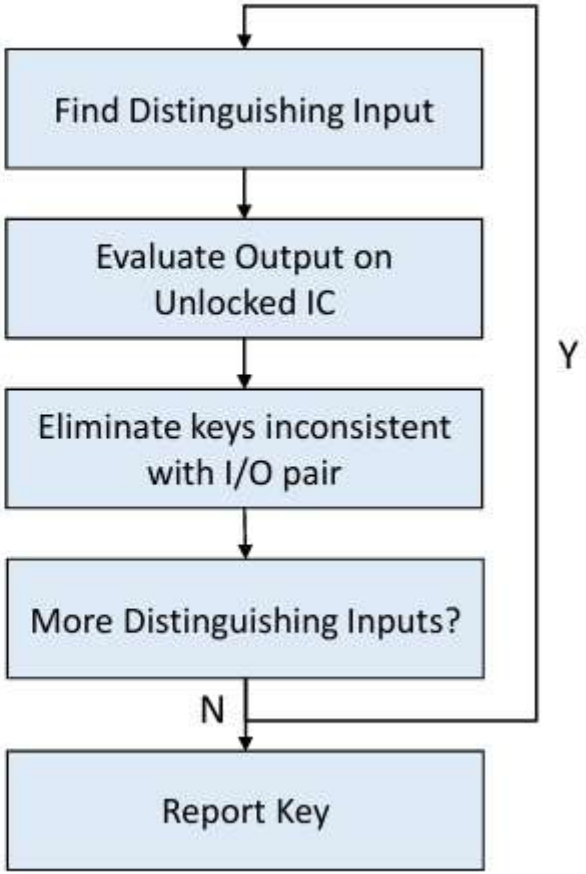


SAT ATTACK

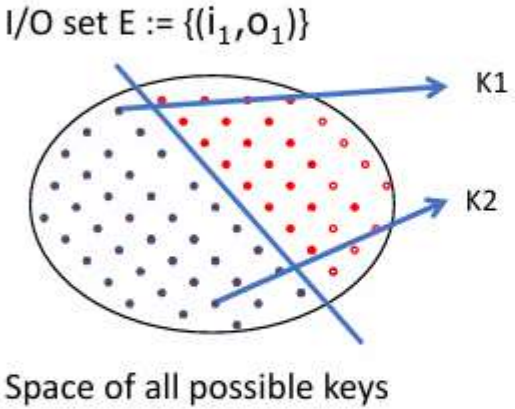
I/O set E := $\{(i_1, o_1)\}$

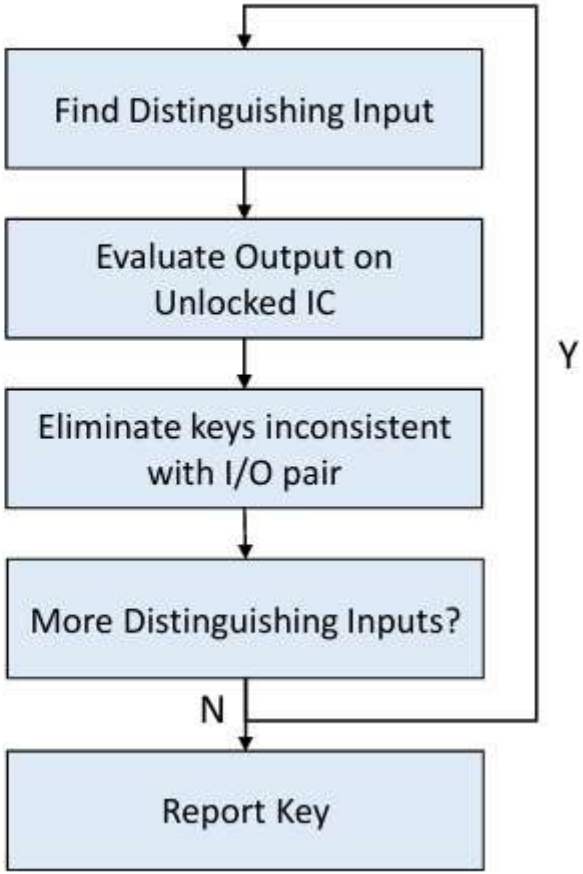


Space of all possible keys

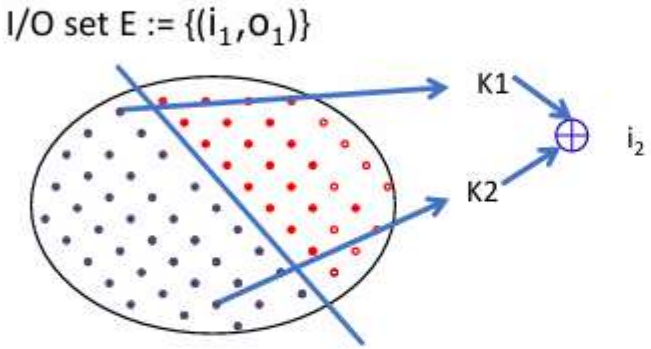


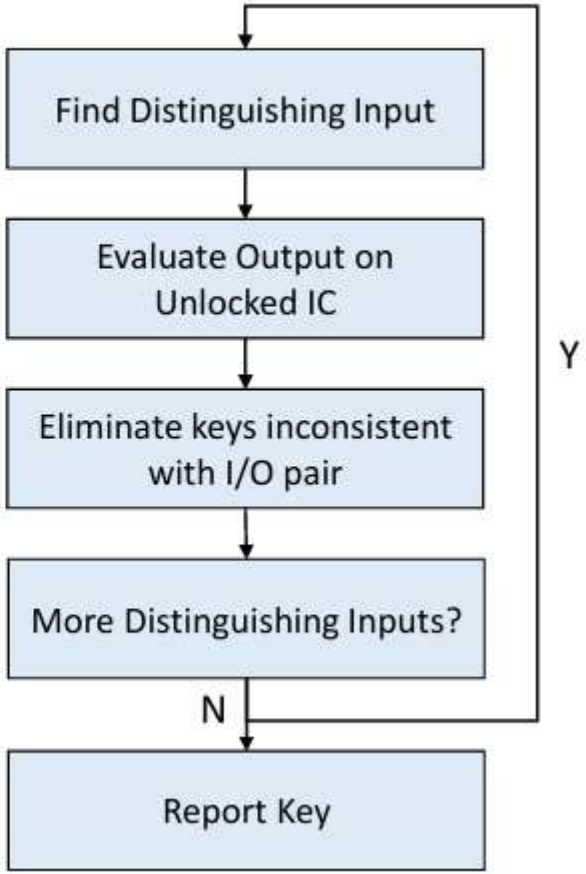
SAT ATTACK


I/O set E := {(i1,o1)}
K1
K2
Space of all possible keys

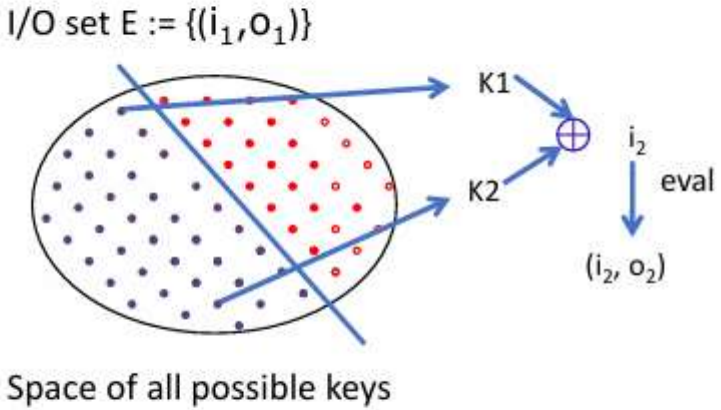

Find Distinguishing Input
Evaluate Output on Unlocked IC
Eliminate keys inconsistent with I/O pair
More Distinguishing Inputs?
Y
N
Report Key

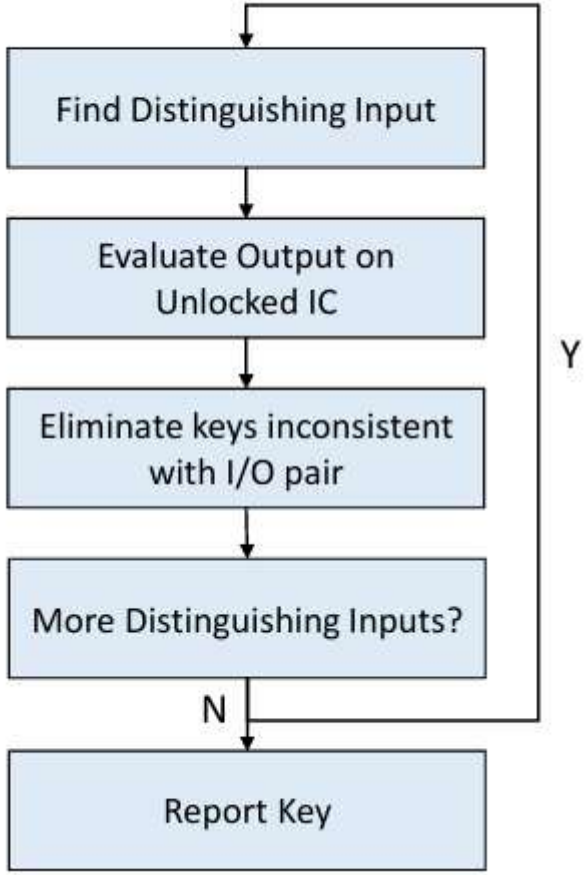
SAT ATTACK


I/O set E := {(i1,o1)}
K1
K2
i2
Space of all possible keys


Find Distinguishing Input
Evaluate Output on Unlocked IC
Eliminate keys inconsistent with I/O pair
More Distinguishing Inputs?
Y
N
Report Key

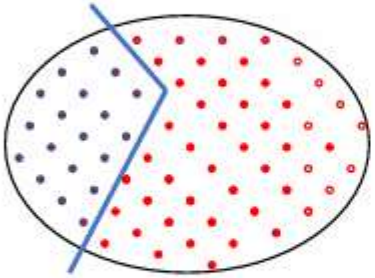
SAT ATTACK


I/O set E := {(i1,o1)}
K1
K2
i2
eval
(i2, o2)
Space of all possible keys

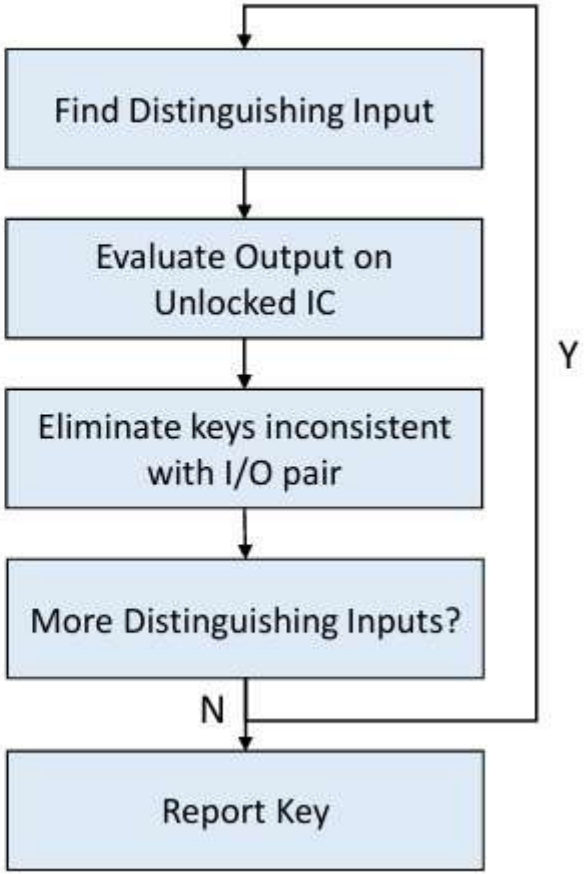

Find Distinguishing Input
Evaluate Output on Unlocked IC
Eliminate keys inconsistent with I/O pair
More Distinguishing Inputs?
Y
N
Report Key

SAT ATTACK

I/O set E := E U $\{(i_2, o_2)\}$

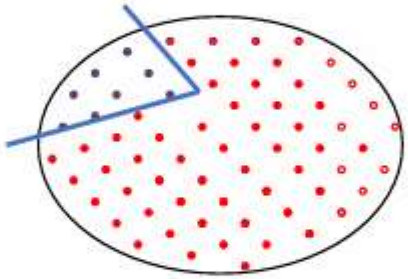


Space of all possible keys

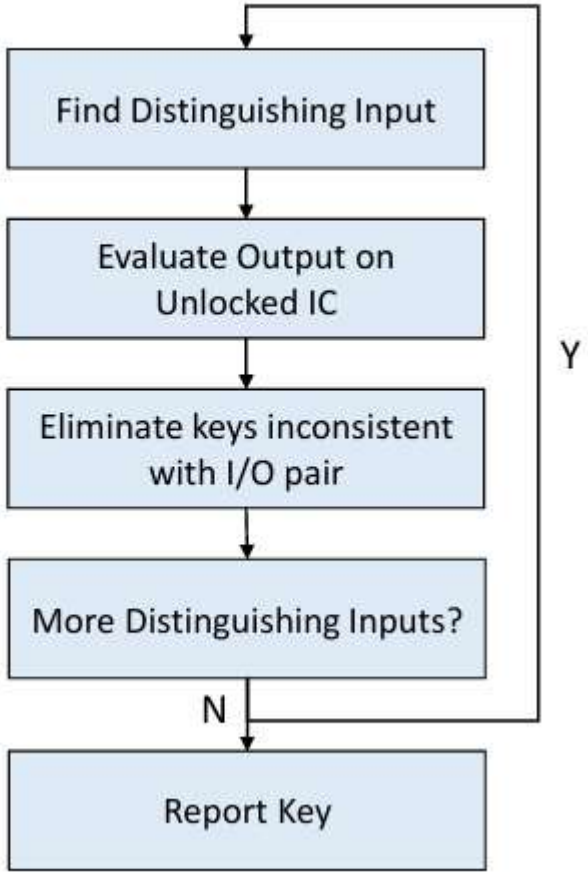


SAT ATTACK

I/O set $E := E \cup \{(i_j, o_j)\}$

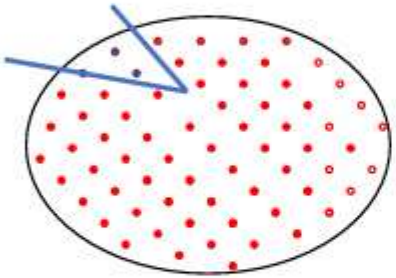


Space of all possible keys

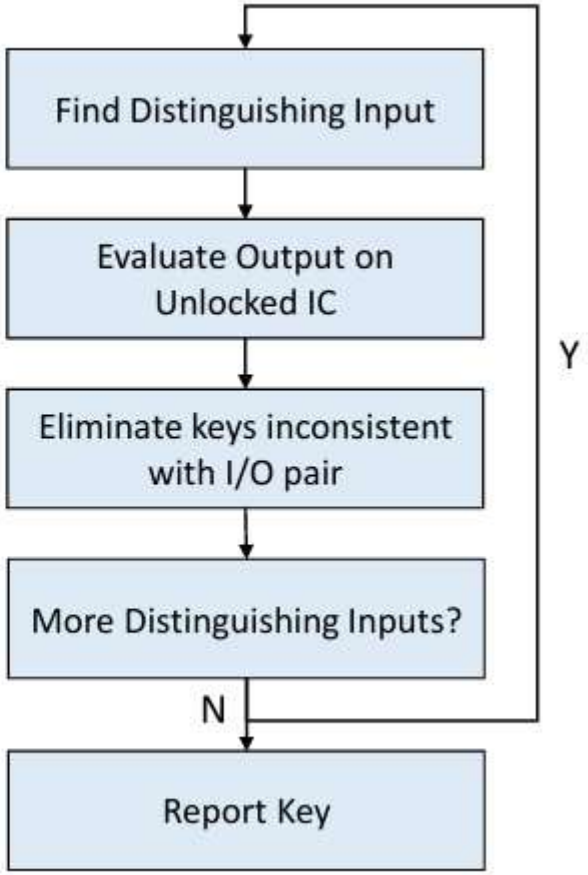


SAT ATTACK

I/O set $E := E \cup \{(i_k, o_k)\}$

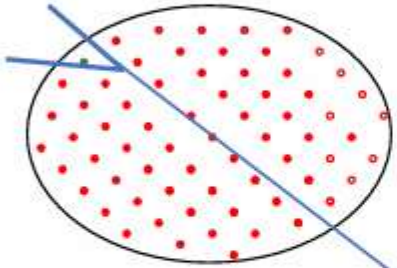


Space of all possible keys

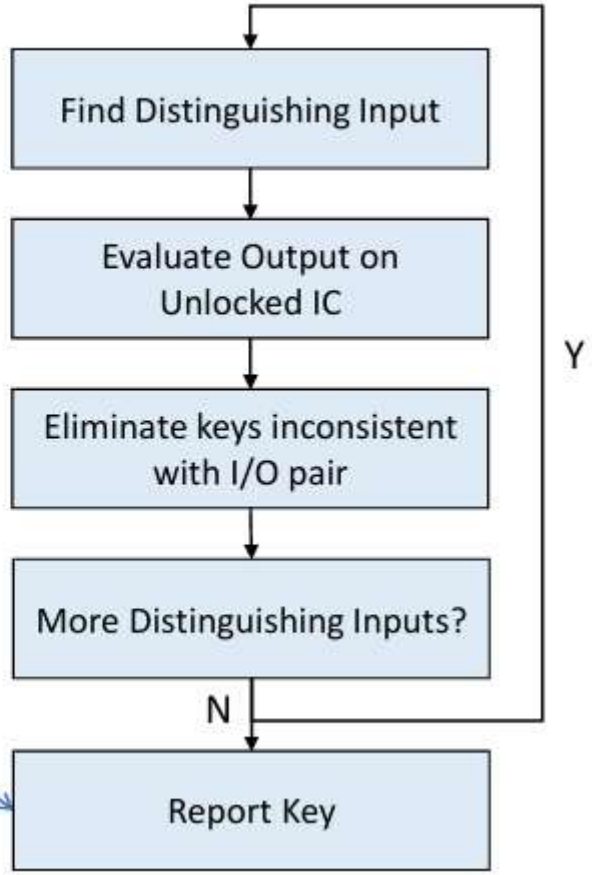


SAT ATTACK

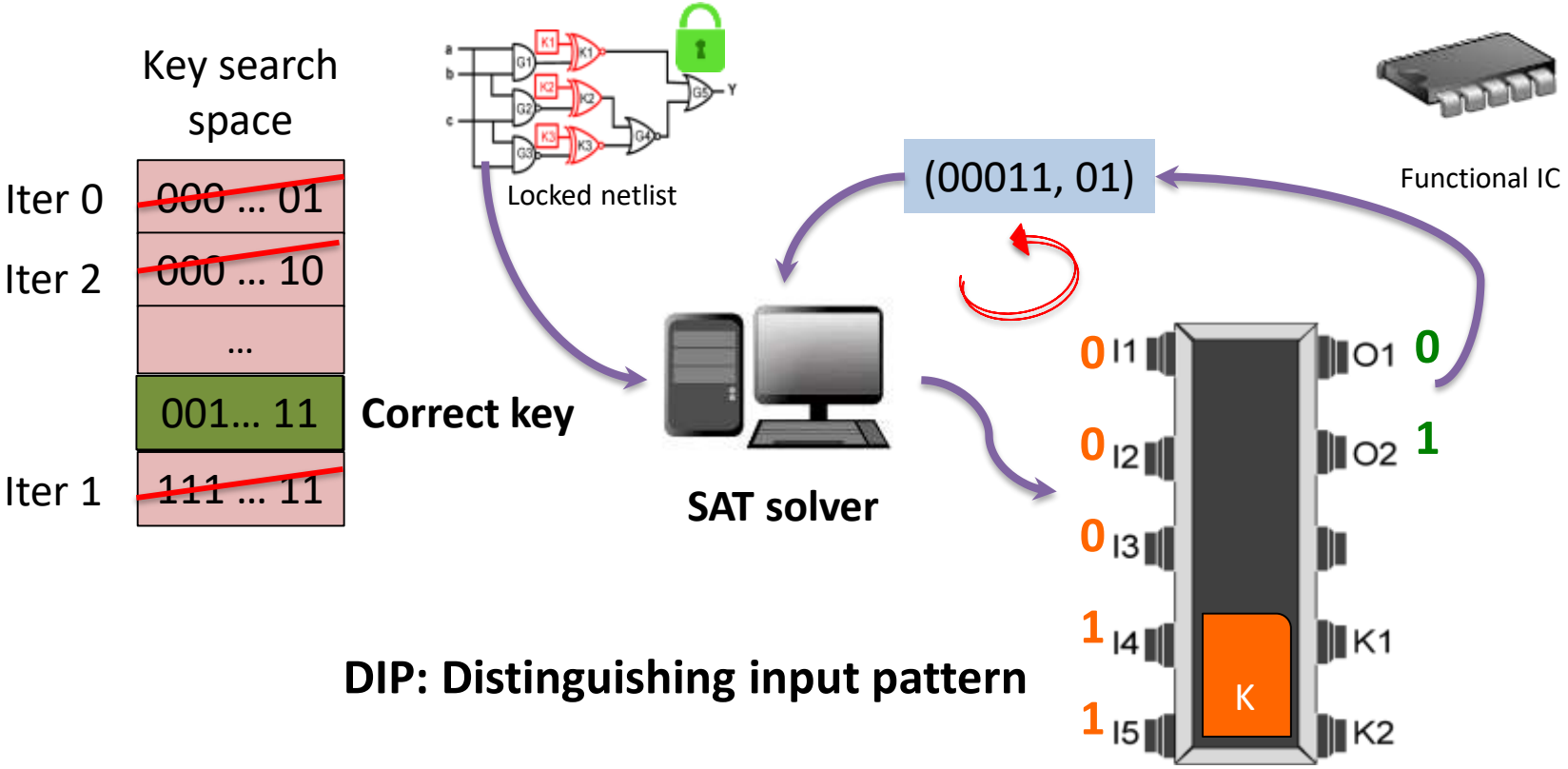
I/O set E := E [$\{(i_n, o_n)\}$



Space of all possible keys



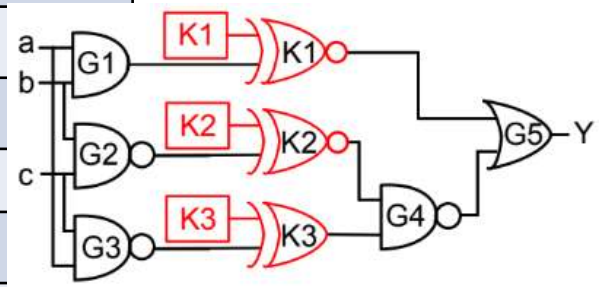
SAT attack

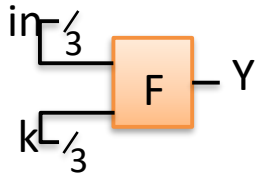


Subramanyan et al. HOST 2015

SAT attack: An example

					Output Y for different key values								
No.	a	b	c	Y	k0	k1	k2	k3	k4	k5	k6	k7	Pruned key values
0	0	0	0	0	1	1	1	1	1	1	0	1	
1	0	0	1	0	1	1	1	1	1	1	0	1	
2	0	1	0	0	1	1	1	1	1	1	0	1	
DIP 1	0	1	1	1	1	1	1	1	0	1	1	1	Iter 1: {k4}
DIP 3	1	0	0	0	1	1	1	1	1	1	0	1	Iter 3: all incorrect
5	1	0	1	1	1	1	1	1	1	1	1	0	
6	1	1	0	1	1	1	0	1	1	1	1	1	
DIP 2	1	1	1	1	1	0	0	1	1	1	1	1	Iter 2: {k1, k2}





Attack success dictated by DIP distinguishing ability

Thwarting SAT attack

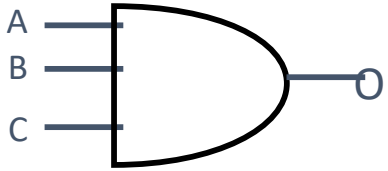
- Worst-case scenario
  - Each DIP eliminates only one key value
- Trade-off
  - SAT attack resilience vs. output corruption

					Output Y for different key values							
No.	a	b	c	Y	k0	k1	k2	k3	k4	k5	k6	k7
0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	1	0	0	0	0
3	0	1	1	1	1	1	1	1	1	1	1	1
4	1	0	0	0	0	0	0	0	0	1	0	0
5	1	0	1	1	1	1	1	1	0	1	1	1
6	1	1	0	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	0

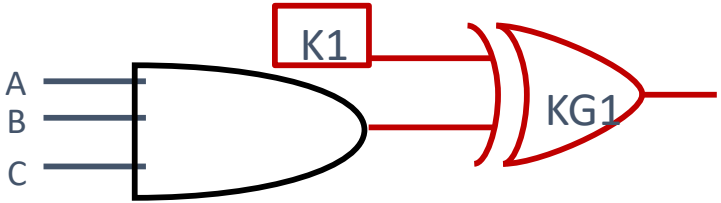
Worst-case # DIPs = 2^k-1

Locked Circuit Practice

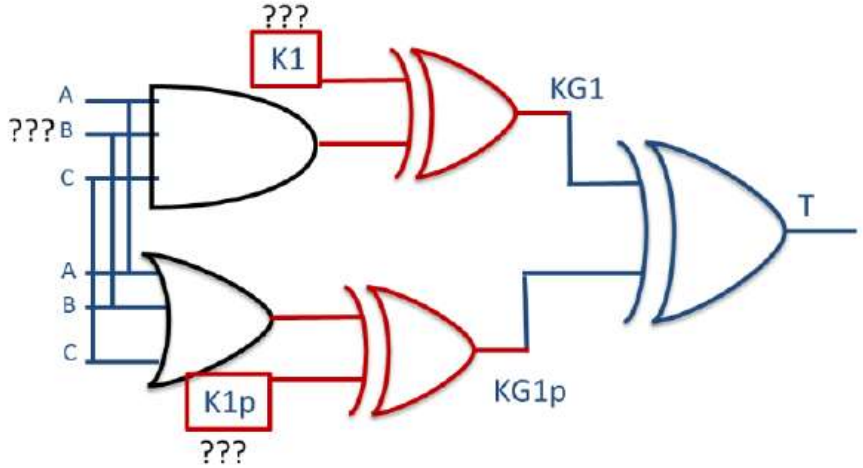
Original Circuit:



Locked Circuit:



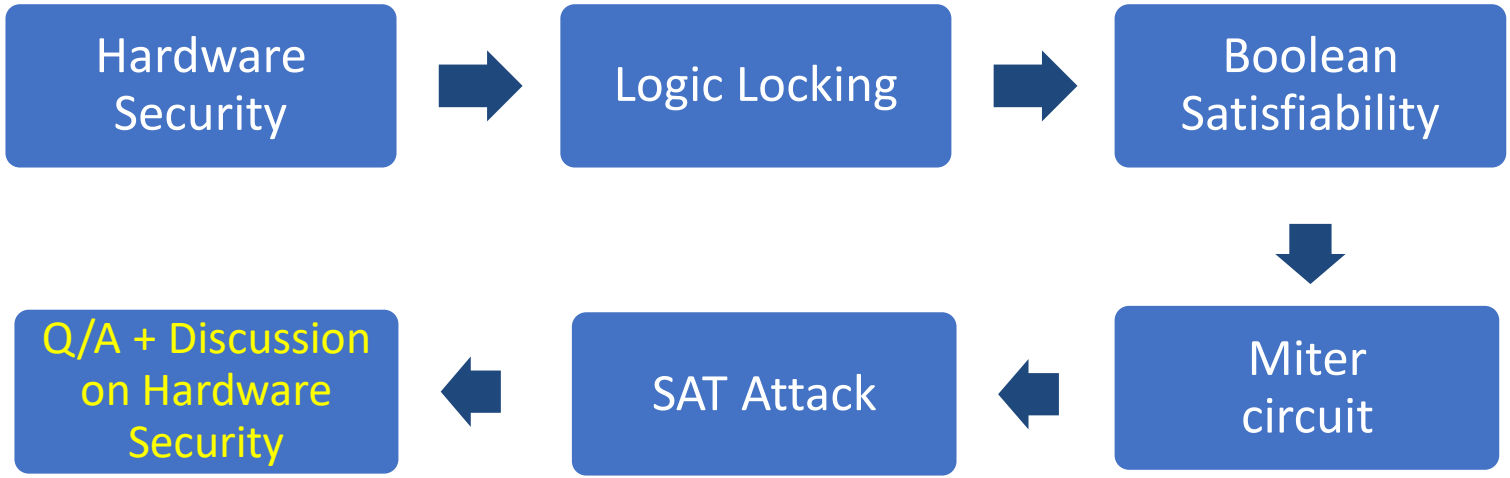
Miter Circuit:



Conclusion

- **Logic Locking:** Prevents IP theft and ensures hardware security.
- **Defends Against Attacks:** Protects circuits from reverse engineering and unauthorized modifications.
- **SAT Attack:** Iteratively eliminates incorrect keys using circuit constraints.
- **SAT Attack Limitations:** Struggles against SFLL and other anti-SAT techniques.
- **Resilience Trade-offs:** Security measures balance performance, power, and area overhead.
- **Future Directions:** Continuous improvements in locking techniques to counter evolving attacks.

Outline


Hardware Security
Logic Locking
Boolean Satisfiability
Miter circuit
SAT Attack
Q/A + Discussion on Hardware Security

Thank You … !!!

Questions

Circuit to CNF Example

Consider a circuit with two AND gates and an OR gate.

Gate 1: c = a ∧ b **Gate 2:** d = x ∧ y **Gate 3:** z = c ∨ d

The CNF encoding represents the circuit's functionality in a format that a SAT solver can understand and process

1 **Gate 1**

c = a ∧ b

2 **Gate 2**

d = x ∧ y

3 **Gate 3**

z = c ∨ d

CNF Clauses:

1. **Gate 1 (AND):**
   $(\neg c \vee a) \wedge (\neg c \vee b) \wedge (c \vee \neg a \vee \neg b).$
2. **Gate 2 (AND):**
   $(\neg d \vee x) \wedge (\neg d \vee y) \wedge (d \vee \neg x \vee \neg y).$
3. **Gate 3 (OR):**
   $(\neg z \vee c \vee d) \wedge (z \vee \neg c) \wedge (z \vee \neg d).$

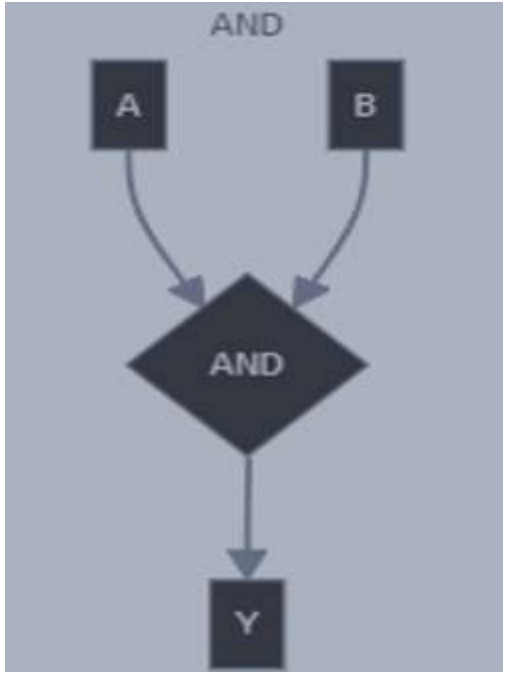
Circuits for Demo and Practice

Simple AND Gate

Original: ($y = a \wedge b$).
Locked: ($y = (a \wedge b) \oplus k$).

CNF Conversion

$(\neg y \vee a) \wedge (\neg y \vee b) \wedge (y \vee \neg a \vee \neg b) \wedge (y \vee \neg k \vee a \vee b) \wedge (\neg y \vee k \vee \neg a \vee \neg b)$



DIMACS CNF Encoding

```
p cnf 3 5
 1  -2  0
 1  -3  0
-1   2  3  0
 1  -4  2  3  0
-1   4 -2 -3  0
```

Circuits for Demo and Practice

Combination of AND and OR Gates:

Original: (y = (a ∧ b) ∨ c).

Locked: (y = (a ∧ b) ∨ c ⊕ k).

CNF Conversion

$(\neg y \vee a) \wedge (\neg y \vee b) \wedge (y \vee \neg a \vee \neg b) \wedge (\neg y \vee c) \wedge (y \vee \neg c) \wedge (y \vee \neg k \vee \neg a \vee \neg b \vee \neg c) \wedge (\neg y \vee k \vee a \vee b \vee c)$

DIMACS CNF Encoding

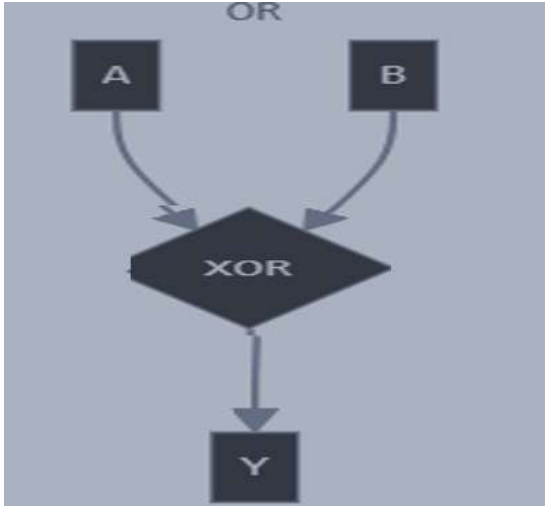
```
p cnf 4 7
 1  -2   0
 1  -3   0
-1  -2  -3   0
 1  -4   0
-1  -4   0
 1  -5  -2  -3  -4   0
-1   5   2   3   4   0
```

Circuit for Quiz

XOR Gate:

Original: ($y = a \oplus b$).

Locked: ($y = (a \oplus b) \oplus k$).



Combination Gate:

Original: ($y = (a \vee b) \vee c$).

Locked: ($y = ((a \vee b) \vee c) \oplus k$).

Miter Circuit Example

Circuit A

$y1 = a \wedge b$

Circuit B

$y2 = a \wedge b$

Miter Output

$z = y1 \oplus y2$