Supply Chain Vulnerabilities of ICs and Mitigation Through Design-for-Trust

Ozgur Sinanoglu

جامعة نيويورك أبوظبي
NYU | ABU DHABI

21st IEEE SMACD Conference

July 10, 2025

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

National Science Foundation WHERE DISCOVERIES BEGIN

TII

SRC Semiconductor Research Corporation

ATIC Advanced Technology Investment Company A Mubadala Company

GLOBAL FOUNDRIES

CENTER FOR CYBER SECURITY

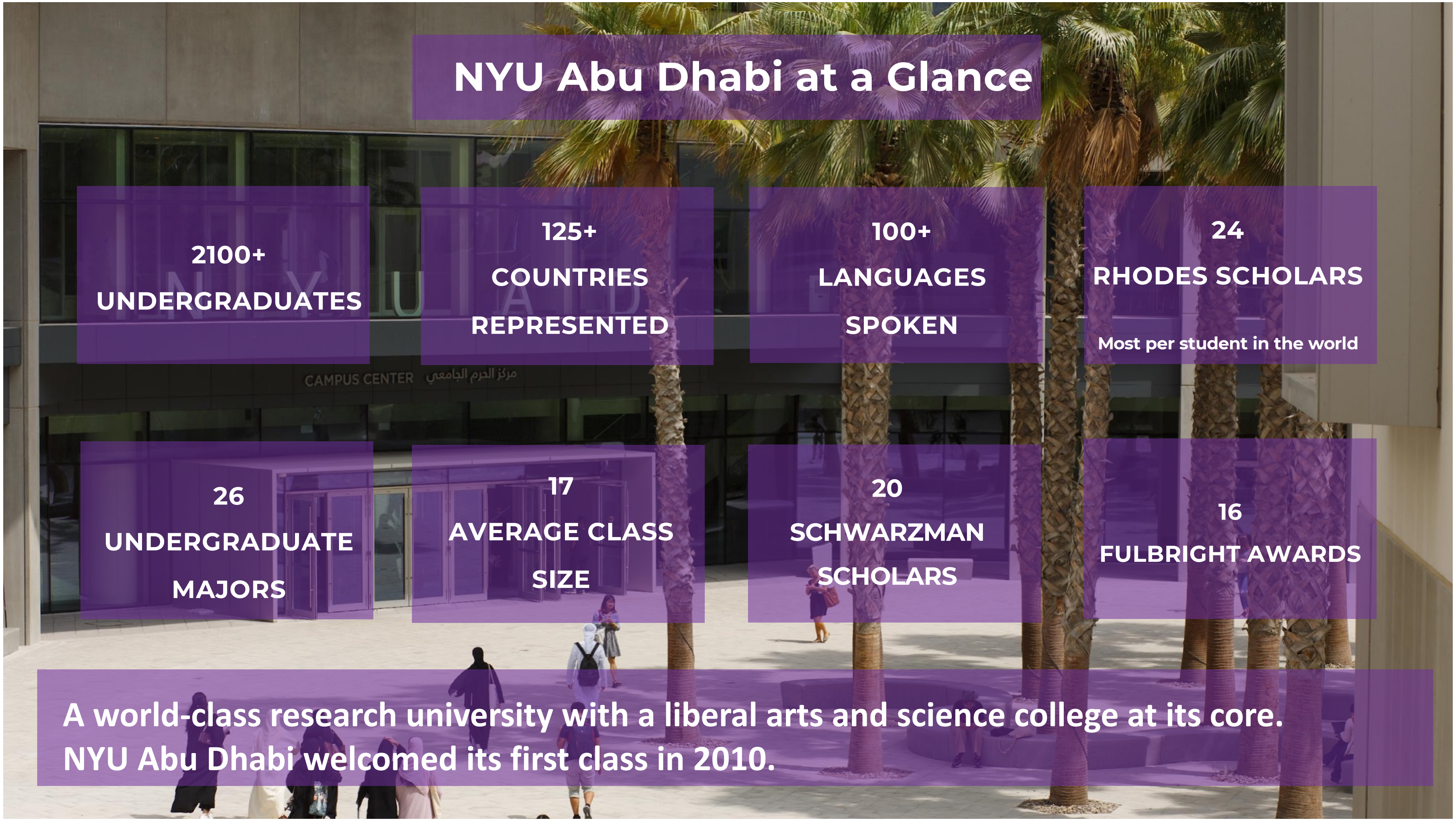
NYU – Global Network University

3 degree-granting campuses

New York, Abu Dhabi (2010-), Shanghai (2012-)

15 global network sites

Accra, Berlin, Buenos Aires, Florence, London, Los Angeles, Madrid, Paris, Prague, Sydney, Tel Aviv, Washington, D.C



NYU Abu Dhabi at a Glance

2100+
UNDERGRADUATES

125+
COUNTRIES REPRESENTED

100+
LANGUAGES SPOKEN

24
RHODES SCHOLARS
Most per student in the world

26
UNDERGRADUATE MAJORS

17
AVERAGE CLASS SIZE

20
SCHWARZMAN SCHOLARS

16
FULBRIGHT AWARDS

A world-class research university with a liberal arts and science college at its core.
NYU Abu Dhabi welcomed its first class in 2010.


Research @NYUAD
~400 FACULTY MEMBERS
11 GLOBAL PHD FELLOWSHIP PROGRAMS
18 RESEARCH CENTERS
Cybersecurity
Astrophysics
AI & Robotics
Genomics
Arabic Literature
Public Health
Water
...
7000+ RESEARCH PUBLICATIONS
SINCE 2010
3 NOBEL LAUREATES
#1 IN UAE: WORLD'S TOP SCIENCE JOURNALS
~250 PATENTS FILED AND ~$55M EXTERNAL GRANT AWARDS
CAMPUS CENTER مركز الحرم الجامعي
A world-class research university with a liberal arts and science college at its core.
NYU Abu Dhabi welcomed its first class in 2010.

CENTER FOR
CYBER SECURITY

جامعة نيويورك أبوظبي
NYU | ABU DHABI

- Research of global significance & local relevance
- Educate the next generation of cybersecurity professionals
- Shape public discourse by bringing together cyber security constituencies

Center for Cybersecurity (CCS) @NYUAD

5 AD-based, 2 NY-based core faculty

4 AD-based affiliated faculty

50+ current researchers
- **2 professional chip designers**

17 past researchers placed into universities and industry:
- E.g., Profs at KU Leuven and KAUST, engineers at Qualcomm, and Intel.

Research funded by ADEK, DARPA (US), Intel, NSF (US), Google, TII, etc.

<u>CCS domains of expertise:</u>
- Communication/information security
- Machine learning security
- Privacy enhancing technologies
- **Trusted hardware design**

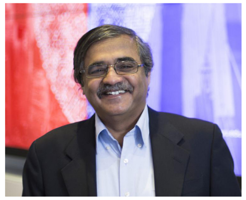
Ozgur Sinanoglu
Principal Investigator,
Director

Michail Maniatakos
Co-Principal Investigator,
Co-director

Christina Pöpper
Co-Principal Investigator,
Co-Director

Muhammad Shafique
Co-Principal Investigator

Sandra Siby
Co-Principal Investigator



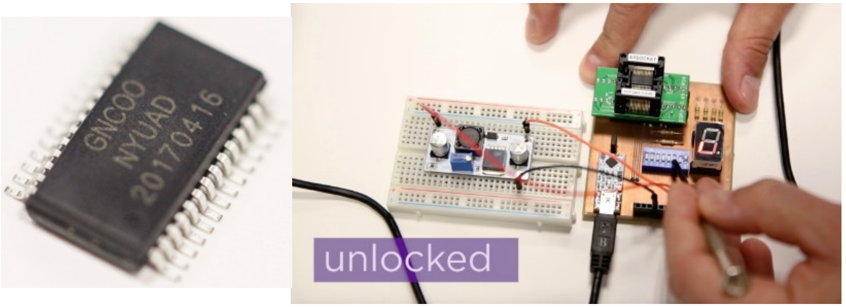
Nasir Memon
Co-Principal Investigator

Ramesh Karri
Co-Principal Investigator

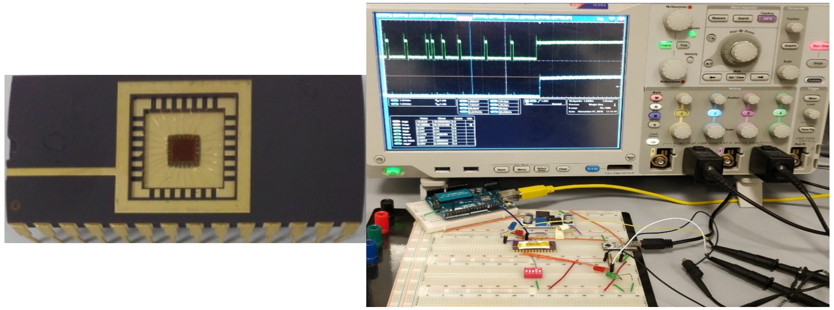
<u>CCS Output (https://sites.nyuad.nyu.edu/ccs-ad/):</u>
- R&D (Publications, Patents, Commercialization Efforts)
- Education & annual hackathon events (Human Capital)

Semiconductor Resources and Capabilities

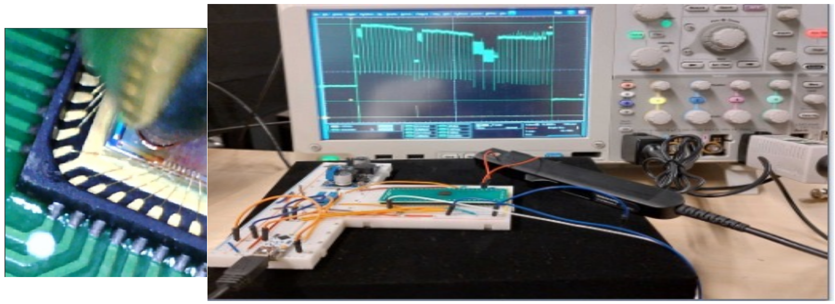
Example prototypes (chip design in-house; fabrication outsourced)



Logic Locked Cortex M0-based microcontroller



Hardware Accelerator for Partial Homomorphic Encryption



Hardware Accelerator for Fully Homomorphic Encryption

So far: Taped out in GF 65nm, GF 55nm, and TSMC 28nm technology
Current: Tape-out with TSMC 22 nm technology in 2025

Supply Chain Vulnerabilities of ICs and Mitigation Through Design-for-Trust

Ozgur Sinanoglu

جامعة نيويورك أبوظبي

NYU | ABU DHABI

21st IEEE SMACD Conference

July 10, 2025

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
NSF National Science Foundation WHERE DISCOVERIES BEGIN
TII
SRC Semiconductor Research Corporation
ATIC Advanced Technology Investment Company A Mubadala Company
SCIENCE TO SHAPE THE FUTURE OF THE ARMY ARMY RESEARCH OFFICE 1951
GLOBAL FOUNDRIES
CENTER FOR CYBER SECURITY

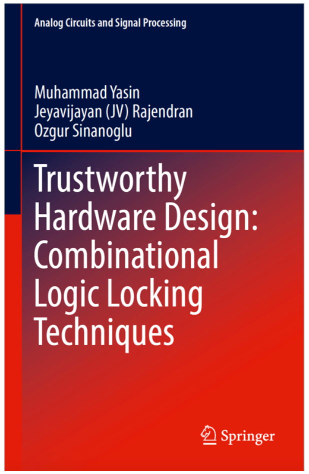
Outline

Part I:

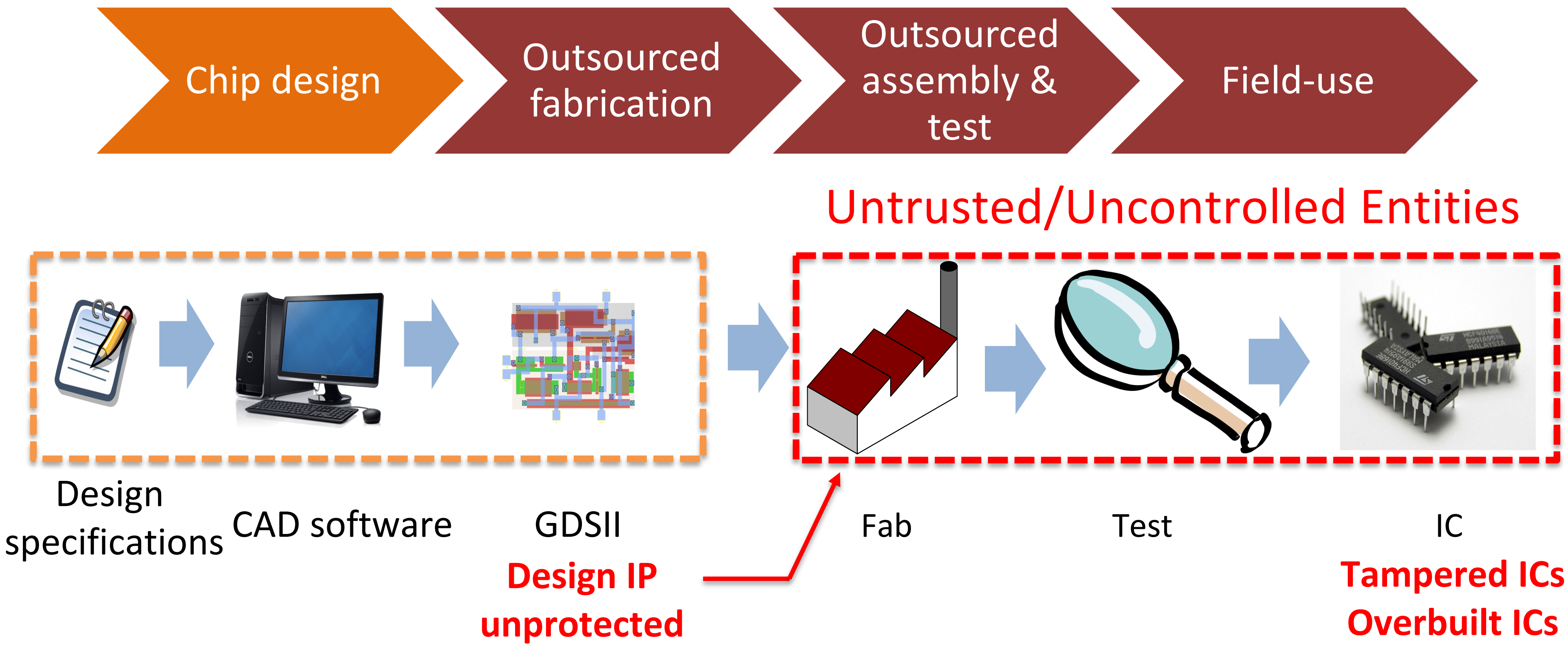
- Security threats for ICs
- Logic locking as a countermeasure
- Lessons learnt and metrics in logic locking
- Unpleasant trade-offs

Part II:

- Re-thinking logic locking
- Future directions



Distributed IC Design and Manufacturing Flow



Problems

1. Chip implementation reveals design details

2. Designers have no control over chip supply chain

IP THEFT

GlobalFoundries vs TSMC (2019)
ASML vs XTAL (2019)
Opticurrent vs Power Integration (2019)
TSMC vs UMC (2018)

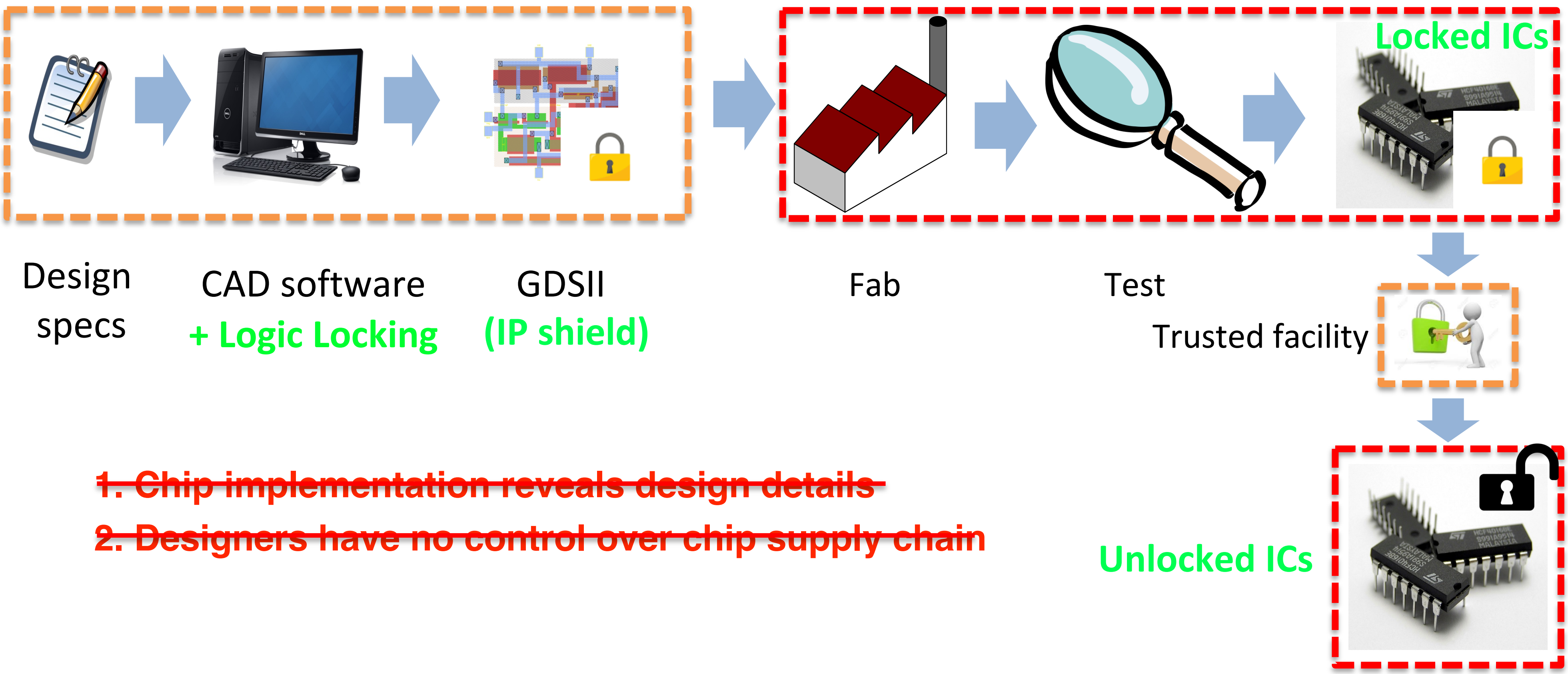
COUNTERFEITING

TI Chips (2019)
CISCO Router (2010)

OVERBUILDING

Anecdotal evidence

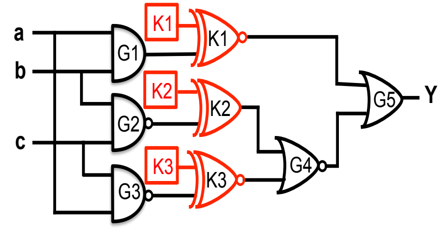
Logic Locking in the Flow

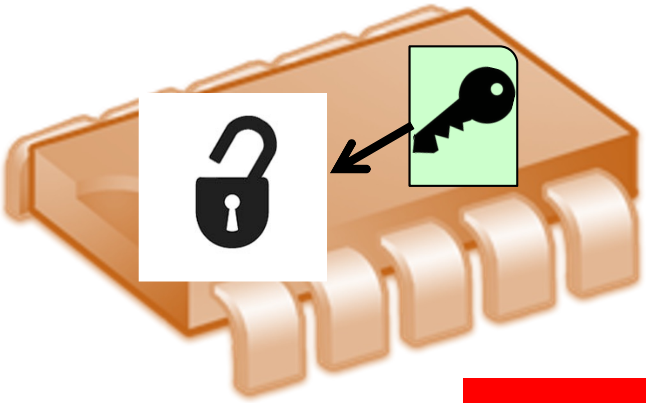


~~1. Chip implementation reveals design details~~

~~2. Designers have no control over chip supply chain~~

Locking and Unlocking Operations





- Logic locking
  - IP owner inserts **locks** into the design
  - Chip **unlocked/activated** by loading the secret key on the chip (one time, NVM)

Secrecy of the key is key!

1. Supply chain control

Chips useless until key is loaded

- Wrong key ➔ Chip fails

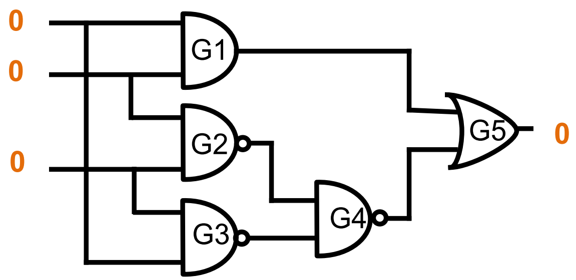
2. Resilience to reverse engineering & piracy

Functionality depends on the **key**

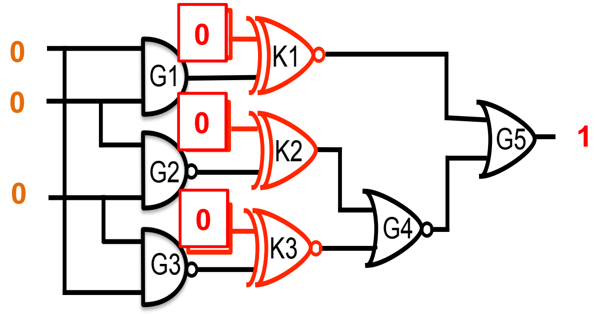
- Gate structure no longer sufficient

Logic Locking - Example

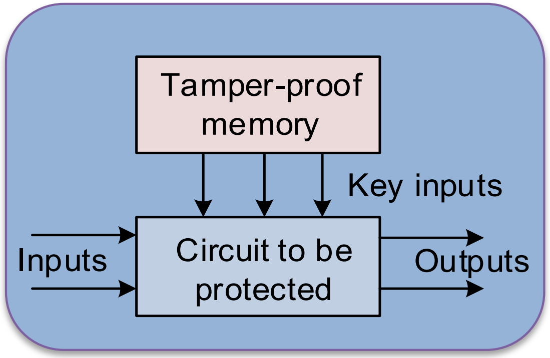
Original circuit



Locked circuit



Correct key: 110



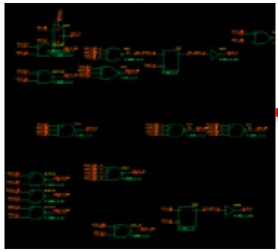
- Chip **unlocked** by loading secret key on tamper

Attacks aim at stealing the key

- **Incorrect** key ➔ **Incorrect** output
- IP owner knows the **secret key**
  - Hidden from everyone else
  - Determines the exact functionality

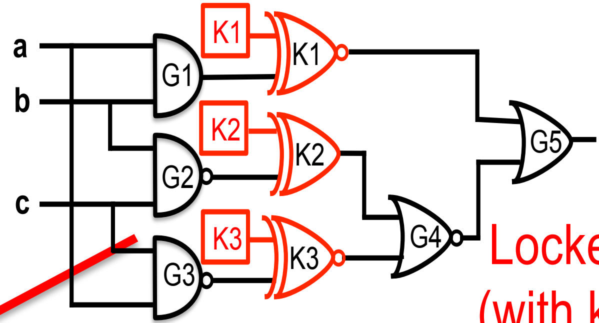
In a nutshell, password-protected chip

Strong Threat Model: Attacks on Logic Locking

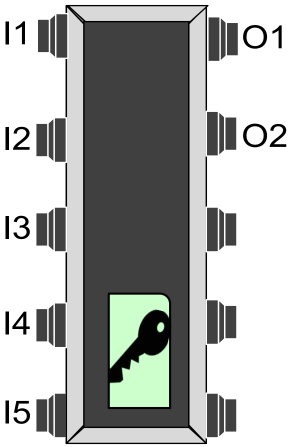


GDSII

Reverse engineering



Locked netlist (with key logic)



Functional IC with key (oracle)

Most powerful attacks

- Simulate netlist to identify (attack) inputs
- Apply inputs to the oracle to get outputs
- Infer the key from input-output pairs

Many attacks: Sensitization attack, SAT attack, etc.

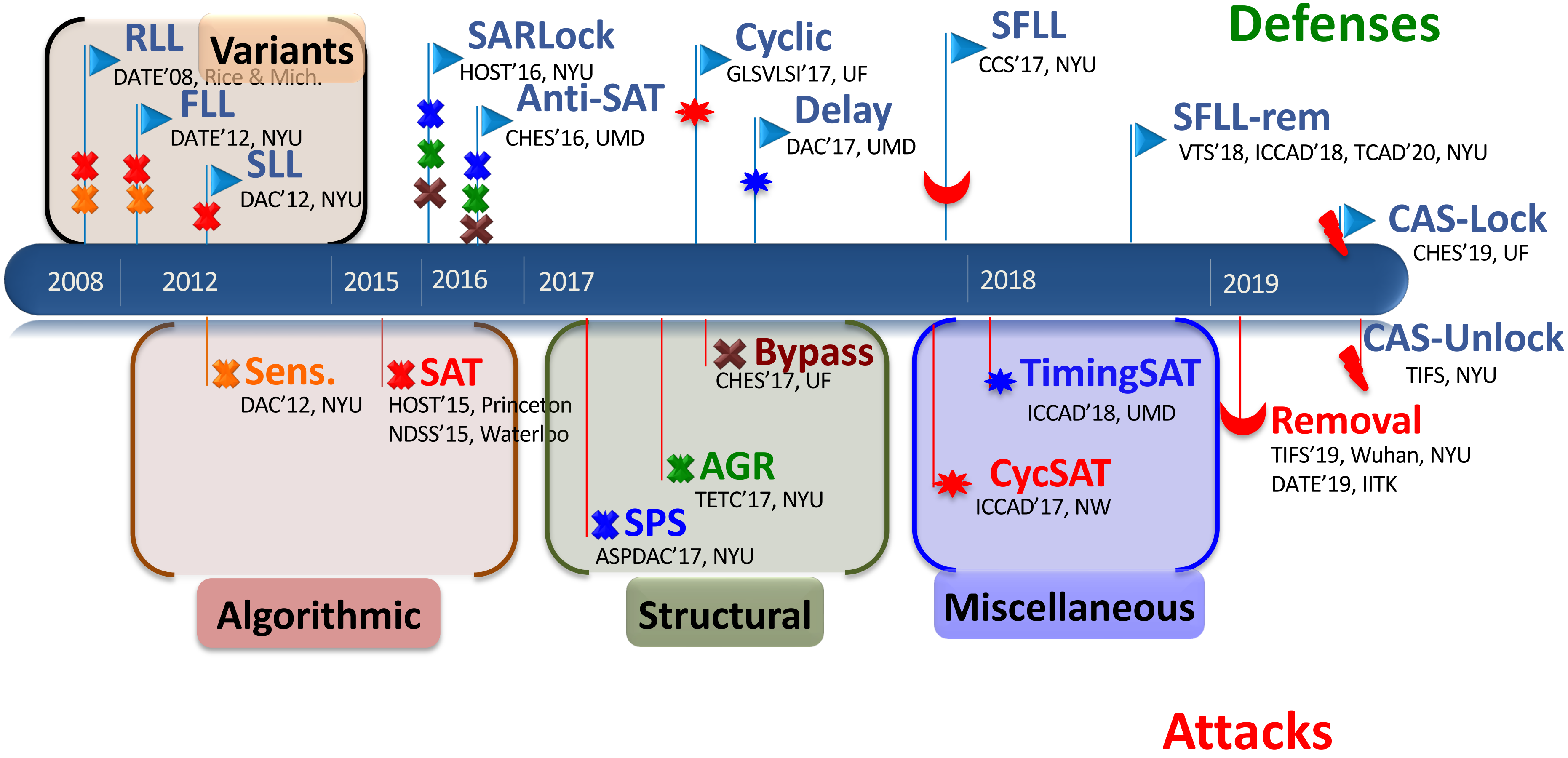
Rajendran, Sinanoglu, et al, DAC 2012

Logic Locking Objectives

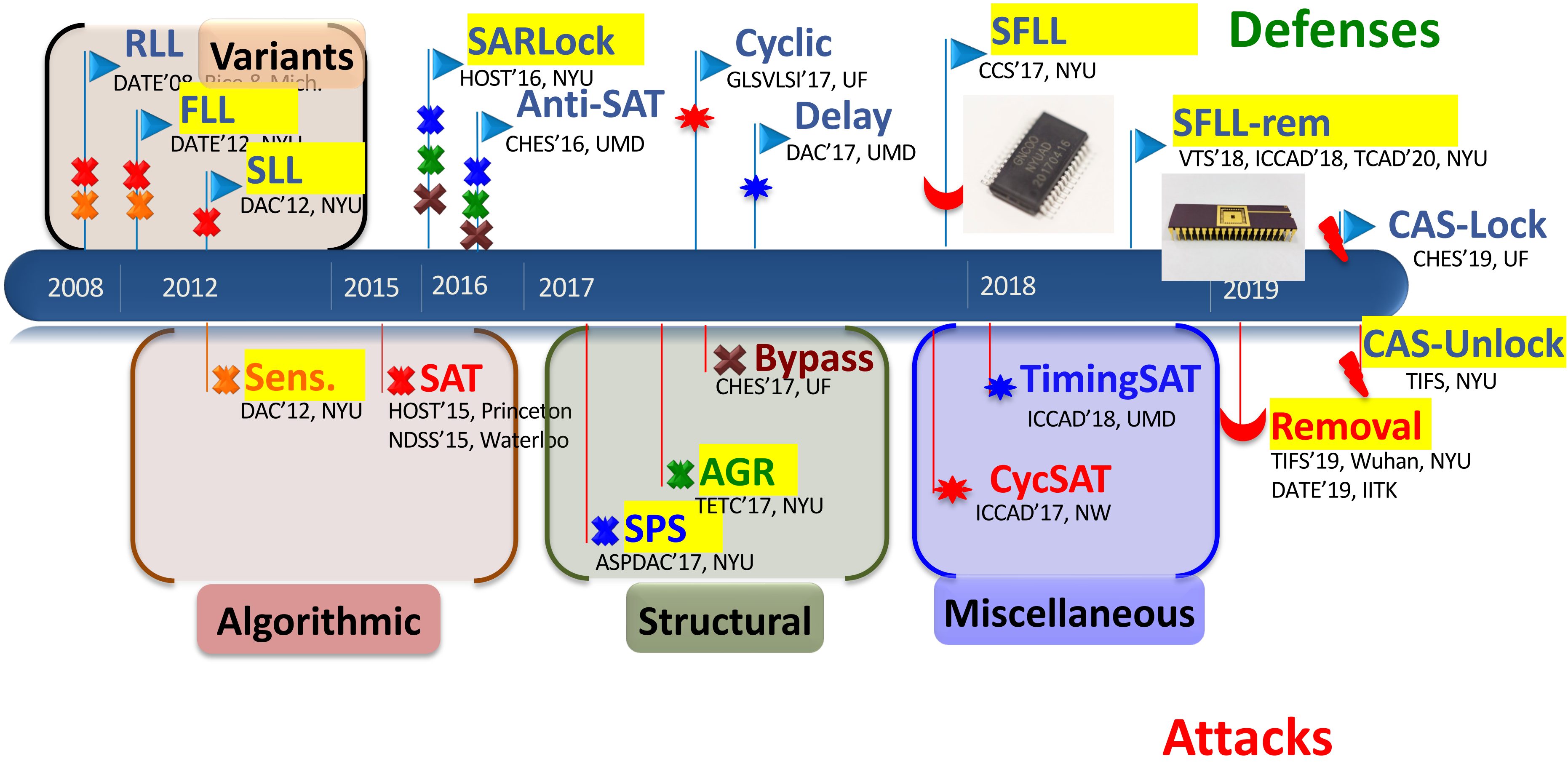
- **Effectiveness** (output corruption):
  - How badly does a locked chip **fail** for an incorrect key?

- **Defense strength**:
  - How **resilient** is the logic locking defense to attacks?

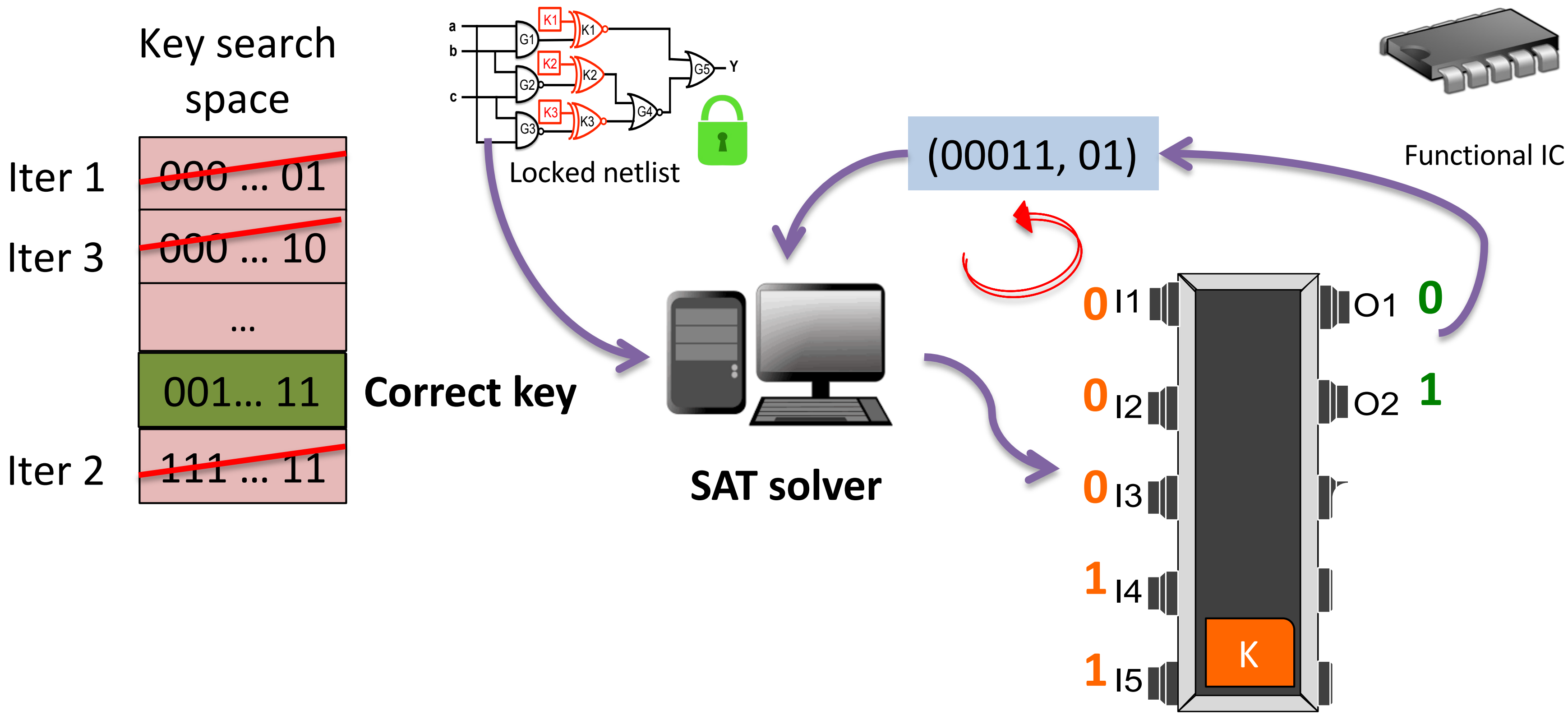
Evolution of Logic Locking (2008 -)



Evolution of Logic Locking (2008 -)



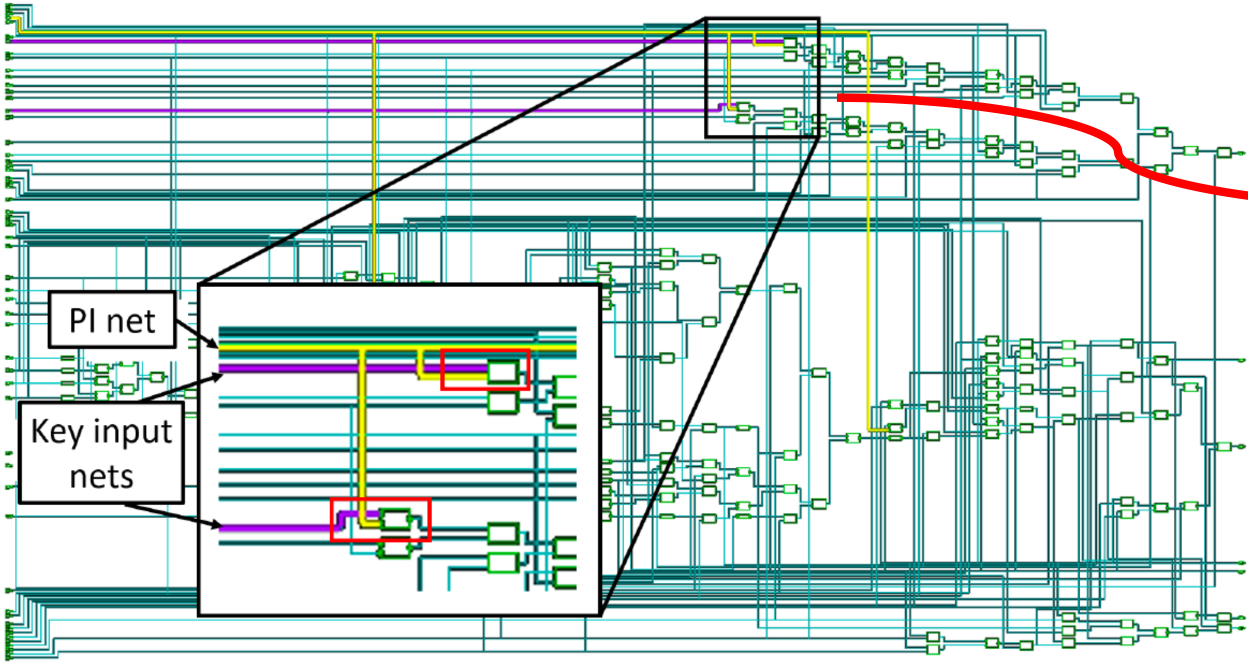
Oracle-based Attacks

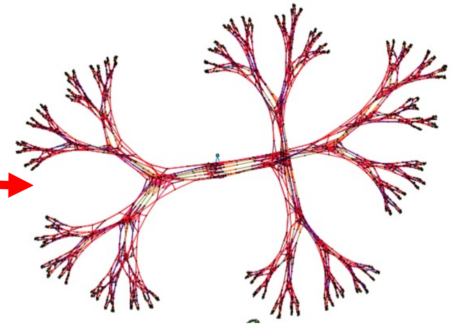


Output corruption helps the attack learn and prune

Subramanyan et al., HOST 2015 El Massad et al., NDSS 2015

Oracle-less Attacks





Analysis of the **locked netlist** to infer the secret key

- Connectivity, signal probability, etc.

Lessons Learnt in 10+ Years

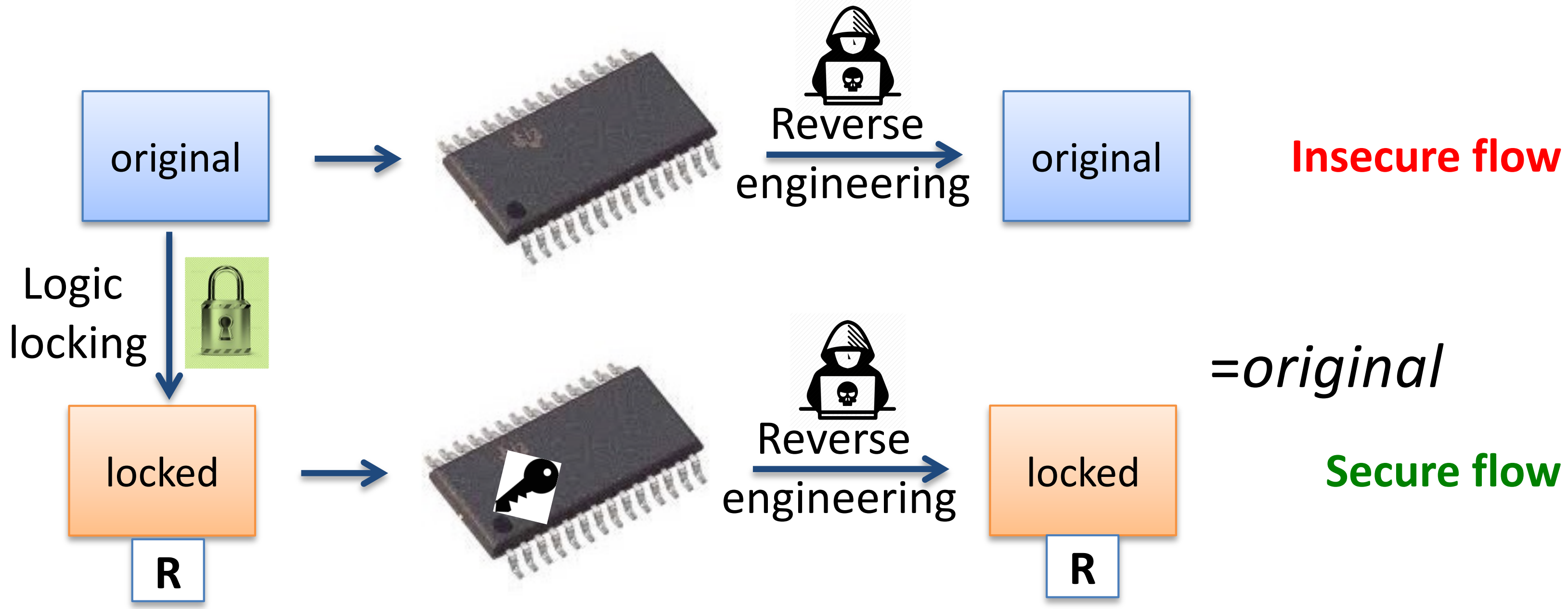
- Oracle (working chip) helps learn from output corruption
  - Early/basic schemes, which were effective, all **broken**
- **Trade off** effectiveness for defense strength?
  - **For good defense strength, effectiveness suffers!**
  - Combining multiple locking defenses won’t help either.

Logic Locking Strategy

1. **Play the trade-off game (effectiveness vs resilience)**

2. Re-define the game

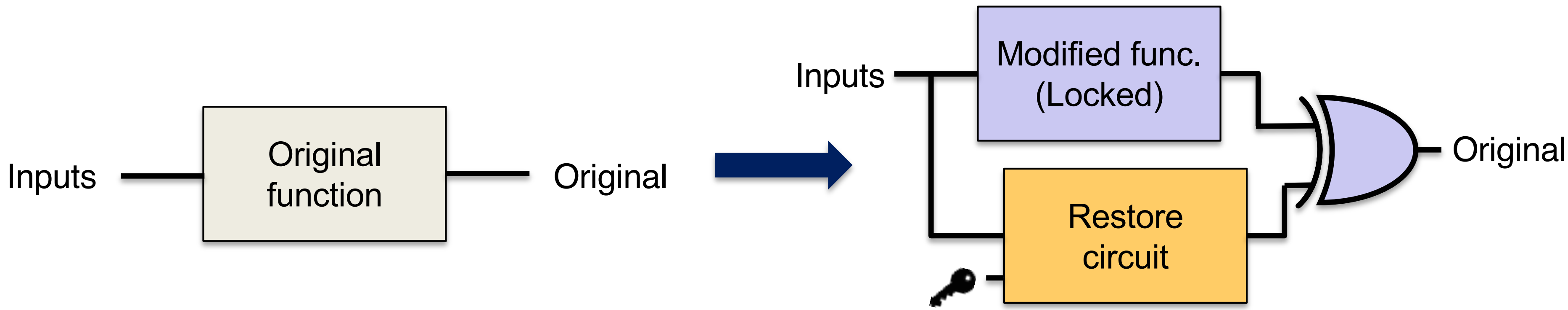
Generic Logic locking for IP Protection

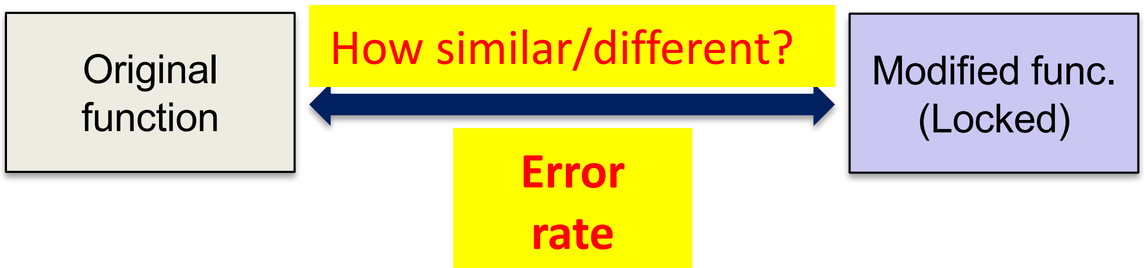


- Logic locking implements ***locked ≠ original*** on-chip
  - The original functionality is restored upon activation (correct key)

Achieving Objectives in Logic Locking

- Effectiveness (output corruption)
- Defense strength (resilience)





Sengupta, … Sinanoglu, TCAD 2020

Security Metric: **Error Rate (ER)**

Error rate: # of input patterns for which modified and original IP differ

- **Low ER:**
  - Oracle-resilient (defense strength)
  - Locked chip with a wrong key almost works fine
- **High ER:**
  - Locked chip useless as black box
  - Vulnerable to oracle attacks

Unpleasant trade-off: Effectiveness vs Defense Strength

Sengupta, … Sinanoglu, TCAD 2020

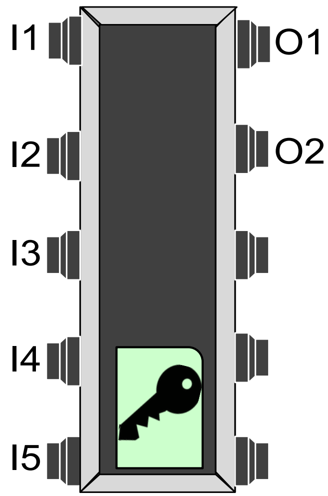
Logic Locking Strategy

1. Play the trade-off game (effectiveness vs resilience)

2. **<u>Re-define the game</u>**

How to Snap Out of This Trade-off?

- Do we have to trade **resilience** for **effectiveness**?
  - **Oracle** helps attacks learn from output corruption



Functional IC with key inside (oracle)

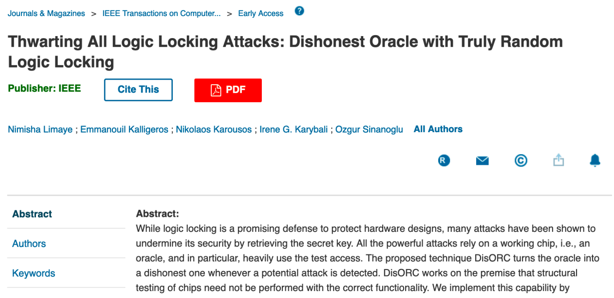
Outline

Part I:

- Security threats for ICs
- Logic locking as a countermeasure
- Lessons learnt and metrics in logic locking
- Unpleasant trade-offs

Part II:

- Re-thinking logic locking
- Future directions


Journals & Magazines > IEEE Transactions on Computer... > Early Access

Thwarting All Logic Locking Attacks: Dishonest Oracle with Truly Random Logic Locking

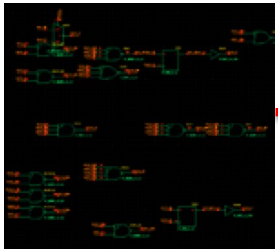
Publisher: IEEE | Cite This | PDF

Nimisha Limaye ; Emmanouil Kalligeros ; Nikolaos Karousos ; Irene G. Karybali ; Ozgur Sinanoglu | All Authors

Abstract | Authors | Keywords

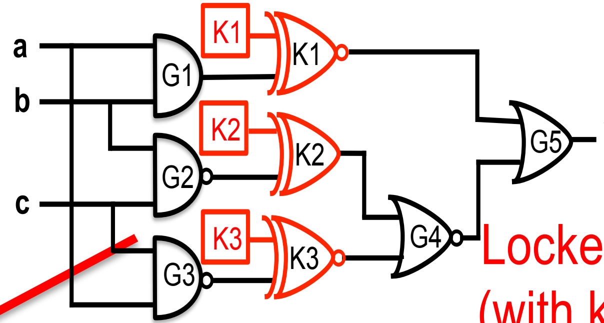
Abstract:
While logic locking is a promising defense to protect hardware designs, many attacks have been shown to undermine its security by retrieving the secret key. All the powerful attacks rely on a working chip, i.e., an oracle, and in particular, heavily use the test access. The proposed technique DisORC turns the oracle into a dishonest one whenever a potential attack is detected. DisORC works on the premise that structural testing of chips need not be performed with the correct functionality. We implement this capability by

Strong Threat Model



GDSII

Reverse engineering



Locked design (with key logic)

Weaker
~~Most powerful~~ attacks

- What if we break the oracle?

✓ **Output corruption no longer a concern**

Rajendran, Sinanoglu, et al, DAC 2012

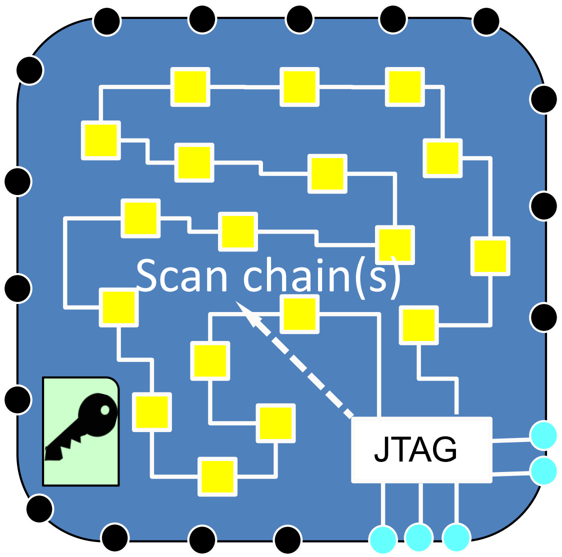
A Cryptex Mechanism?

The Da Vinci Code **Cryptex** (from the movie)

- If one tries to force the cryptex open, the vial will break and the vinegar will dissolve the papyrus (secret message) before it can be read

How to Break the Oracle Without Breaking the IC

- Every IC has scan chains to facilitate test/debug
- Scan chains: Design flops chained together for serial access
- Designs are controlled/observed mostly by scan cells



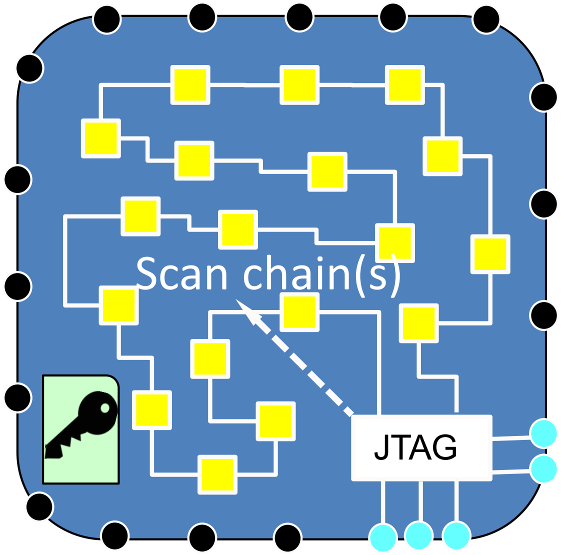
Functional IC with key inside (oracle)

Access to oracle
=
Access to scan chains

Break ~~the scan~~ chains?

How to Break the Oracle Without Breaking the IC

- Every IC has scan chains to facilitate test/debug
- Scan chains: Design flops chained together for serial access
- Designs are controlled/observed mostly by scan cells



Functional IC with key inside (oracle)

Access to oracle
=
Access to scan chains

Scan locking?

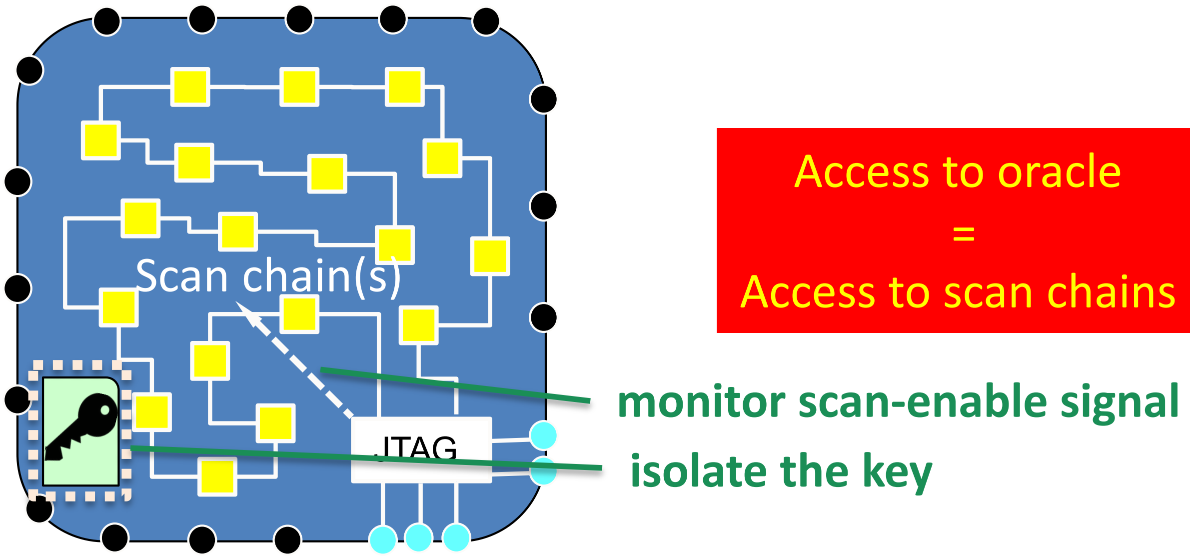
Static, TCAS'20, IIT Kharagpur
Dynamic, TCAD'17, UF
DFS, TVLSI'18, Auburn

Broken!

ScanSAT, TETC'19, NYU
DynUnlock, DATE'20, NYU
Shift&Leak, ICCAD'19, NYU

How to Break the Oracle Without Breaking the IC

- Every IC has scan chains to facilitate test/debug
- Scan chains: Design flops chained together for serial access
- Designs are controlled/observed mostly by scan cells



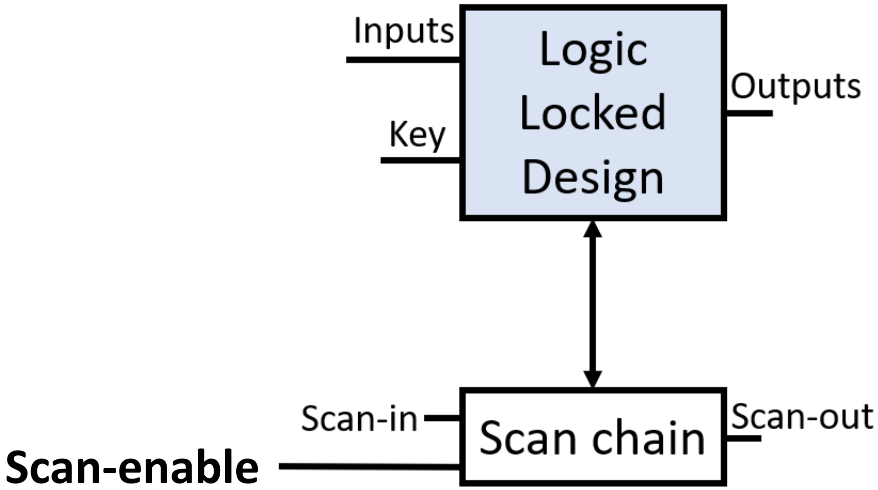
Functional IC with key inside (oracle)

Access to oracle
=
Access to scan chains

1. Detect oracle access
   - Assume it's an attack (paranoia!)
2. Withdraw the key
   - Oracle broken

N. Limaye, ..., O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

Dishonest Oracle (DisORC) Conceptual Architecture

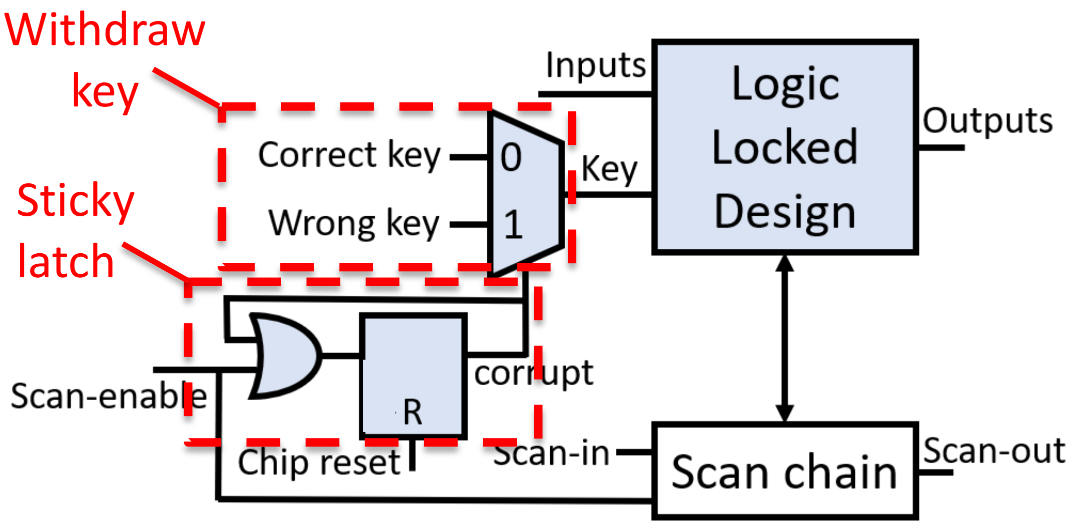


- Attacks use scan chains
- **Detect**: Scan-enable = 1
- **Defense**: Withdraw key until chip reset

Access to scan chains ←→ Scan-enable = 1

N. Limaye, …, O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

Dishonest Oracle (DisORC) Conceptual Architecture



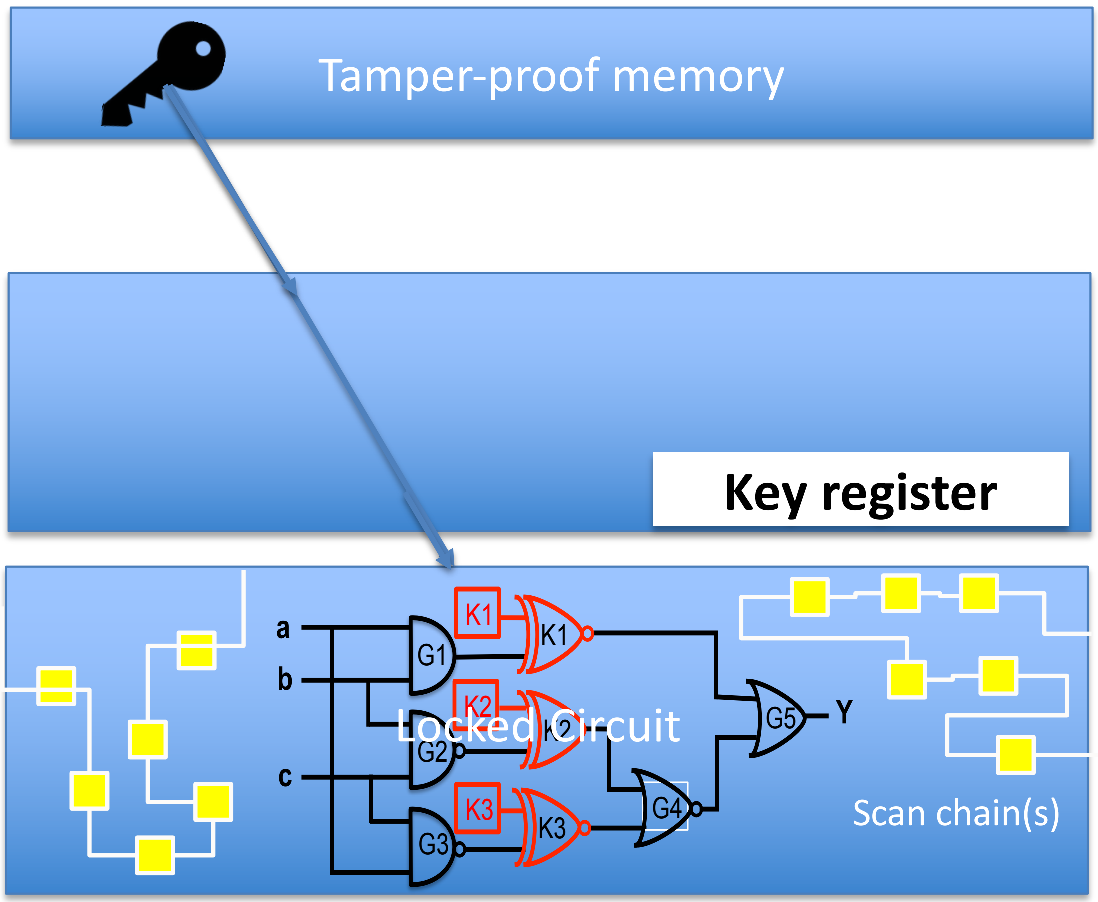
- Attacks use scan chains
- **Detect**: Scan-enable = 1
- **Defense**: Withdraw key until chip reset

Access to scan chains ←→ Scan-enable = 1

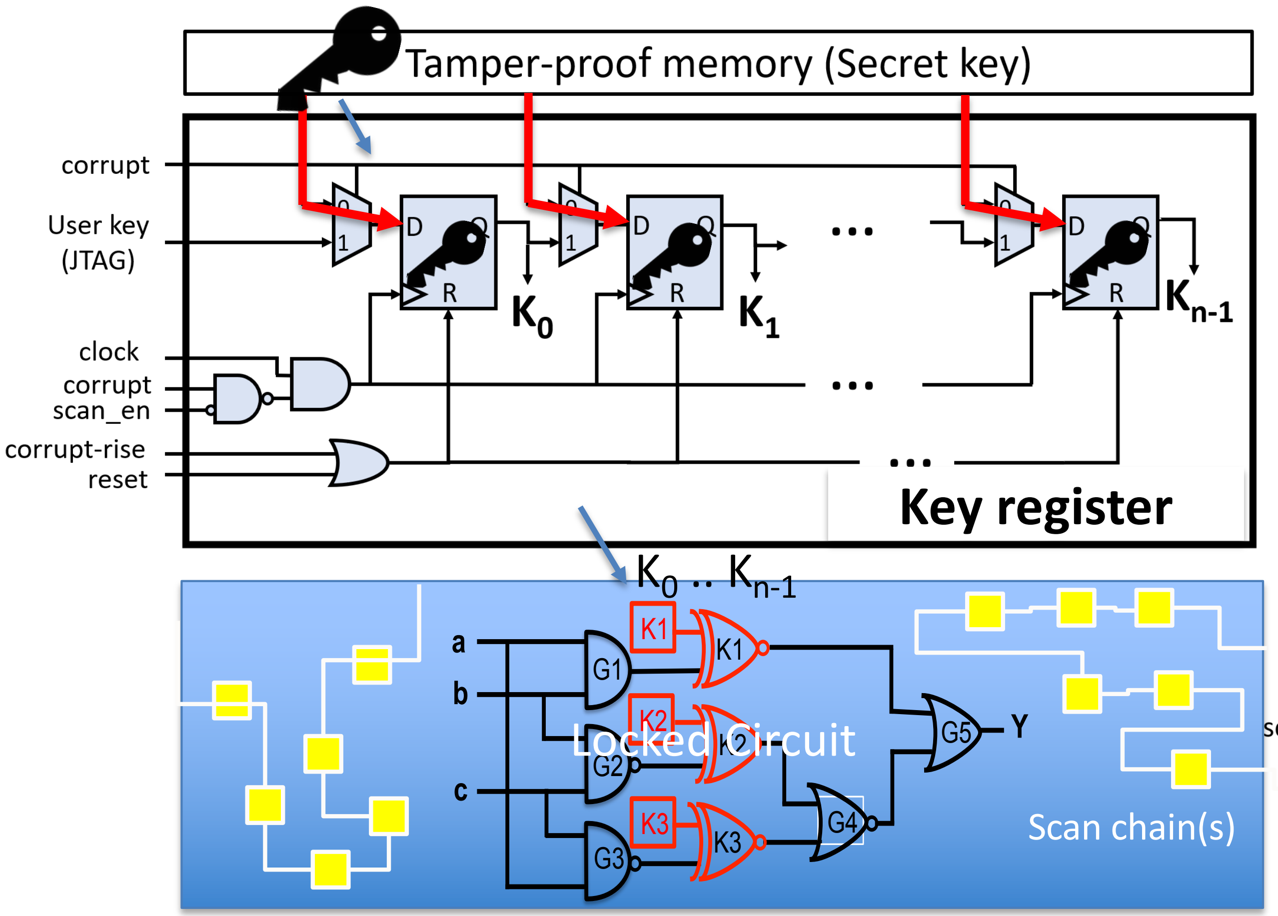
- **Implementation**: Upon scan chain access, erase traces of the key completely
- **Implications** on test, debug?

N. Limaye, …, O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

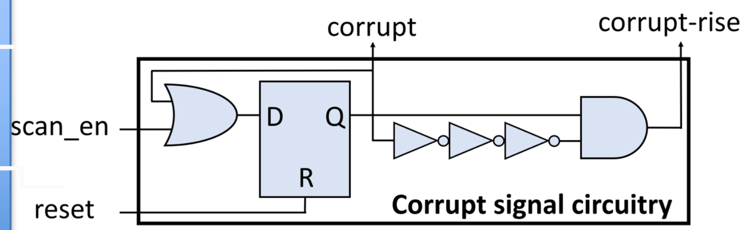
DisORC Implementation



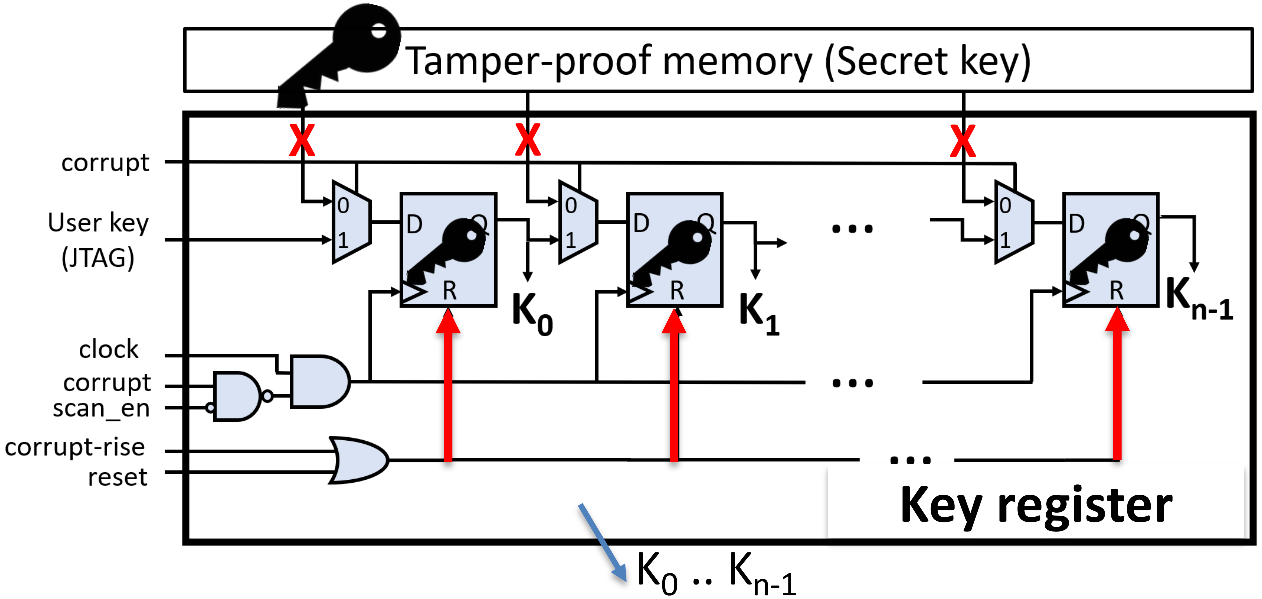
DisORC Implementation



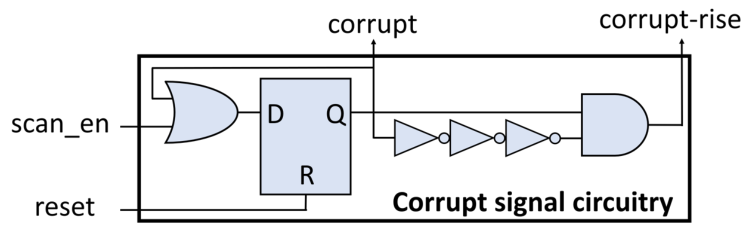
- Clock, scan-enable, and reset are existing signals
- First clock after reset:
  - ➢ 🔑 → key register



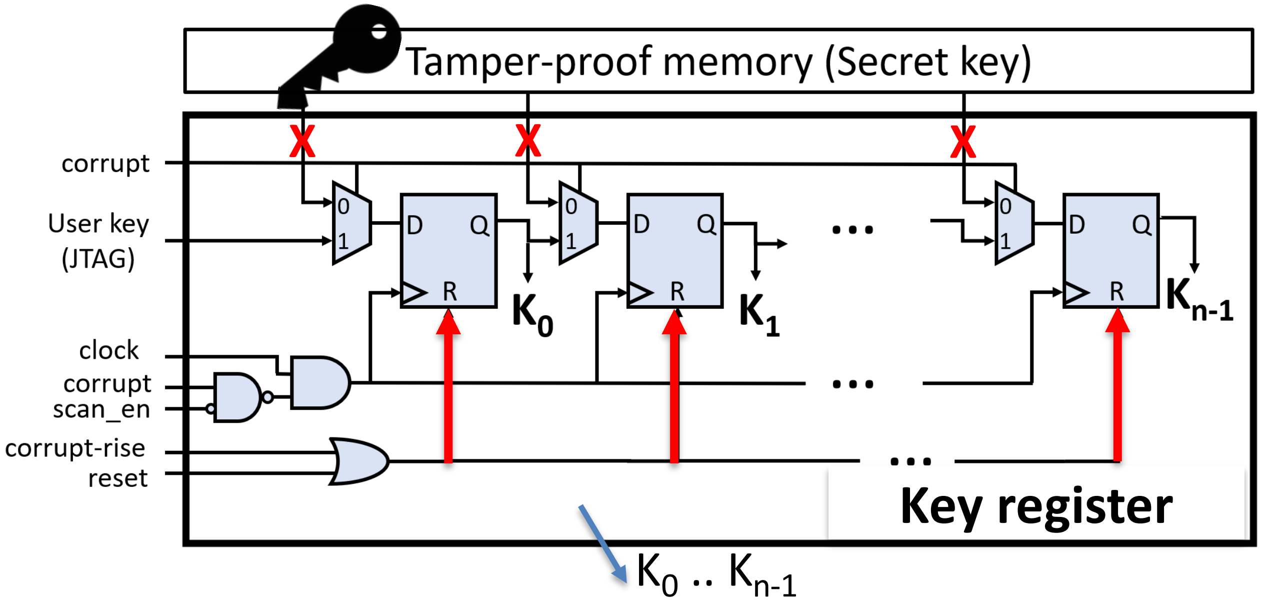
DisORC Implementation



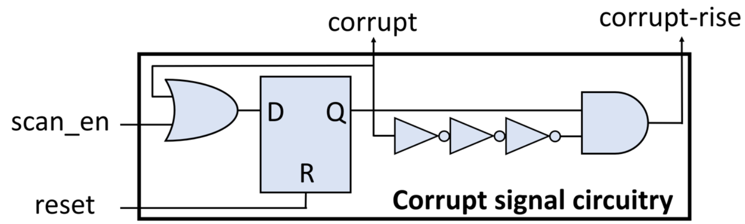
- Clock, scan-enable, and reset are existing signals
- First clock after reset:
  - ➢ 🔑 → key register
- Scan-en = 1:
  - ➢ Corrupt = 1 until reset
  - ➢ Pulse on Corrupt-rise



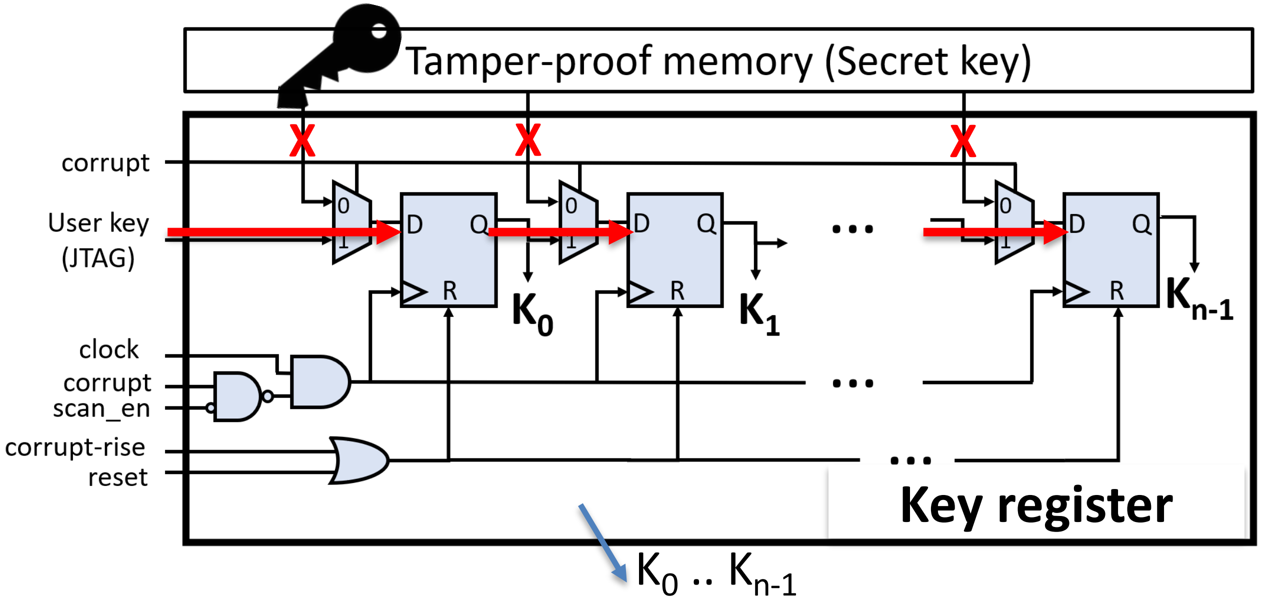
DisORC Implementation



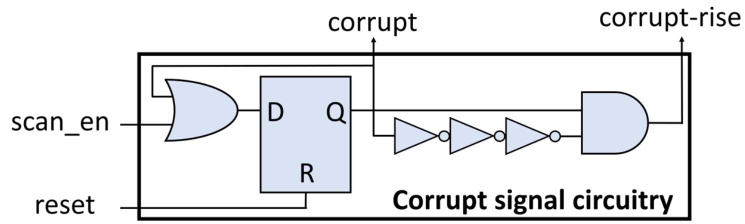
- Clock, scan-enable, and reset are existing signals
- First clock after reset:
  - ➢ 🔑 → key register
- Scan-en = 1:
  - ➢ Corrupt = 1 until reset
  - ➢ Pulse on Corrupt-rise



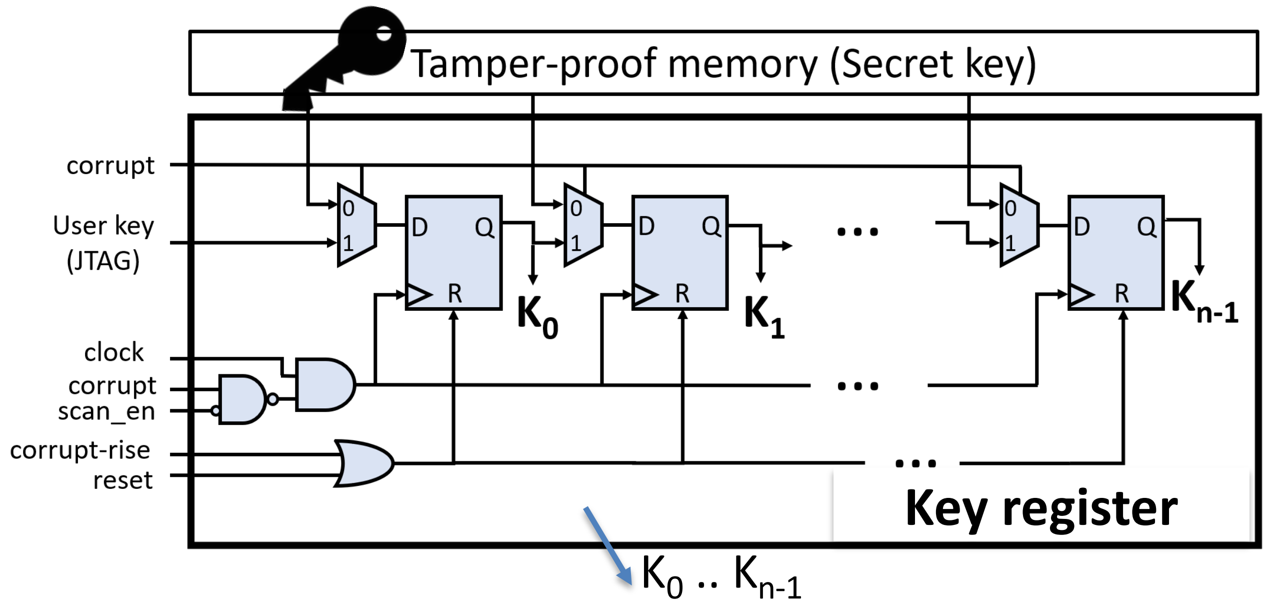
DisORC Implementation



- Clock, scan-enable, and reset are existing signals
- First clock after reset:
  - 🔑 → key register
- Scan-en = 1:
  - Corrupt = 1 until reset
  - Pulse on Corrupt-rise
- "Wrong" key can be loaded by anyone (JTAG)



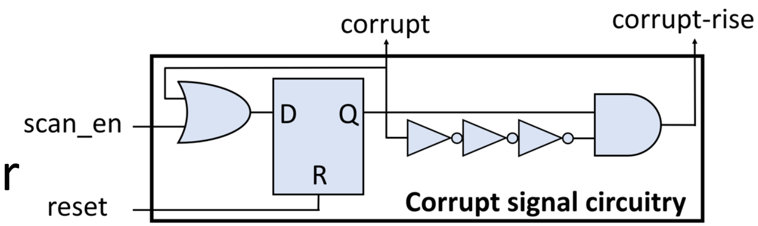
DisORC Implementation



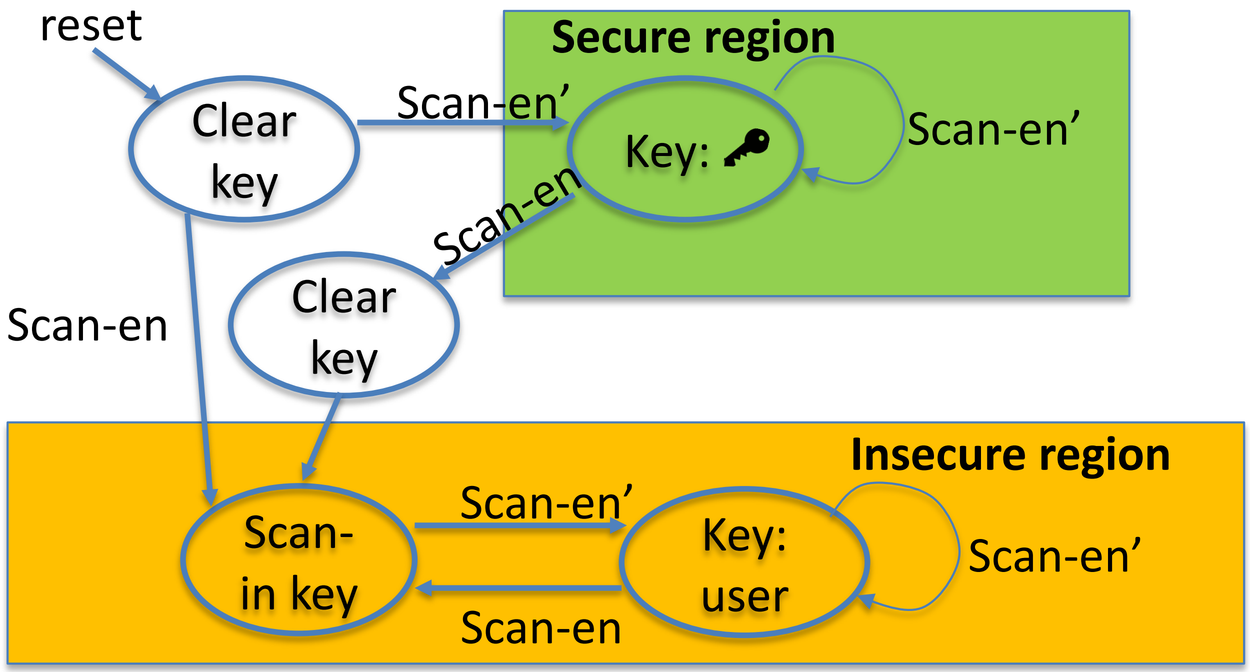
- Clock, scan-enable, and reset are existing signals
- First clock after reset:
  - ➢ → key register
- Scan-en = 1:
  - ➢ Corrupt = 1 until reset
  - ➢ Pulse on Corrupt-rise
- "Wrong" key can be loaded by anyone (JTAG)

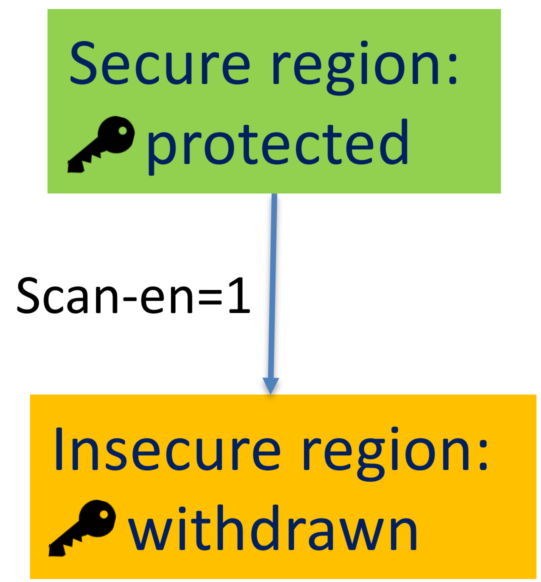
<u>Access to scan chains</u>:

- Immediate reset of key-register
- Immediate disconnection of key from key register
- Traces of key **erased**



DisORC Implementation

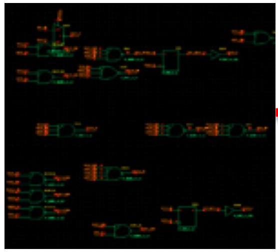




<u>Access to scan chains</u>:

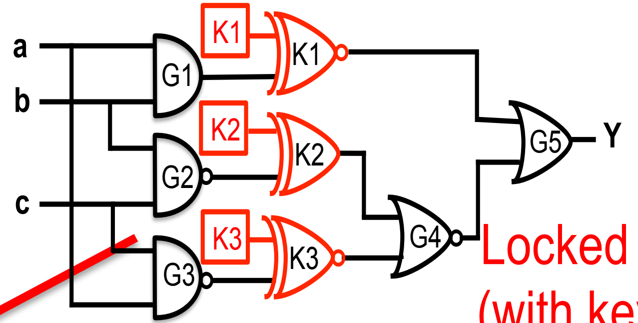
- Immediate reset of key-register
- Immediate disconnection of key from key register
- Traces of key **erased**

Strong Threat Model

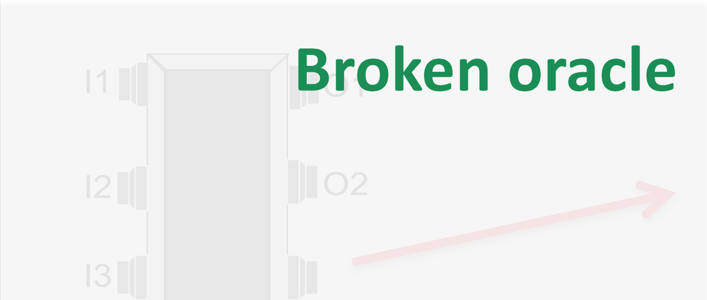


GDSII

Reverse engineering



Locked design (with key logic)



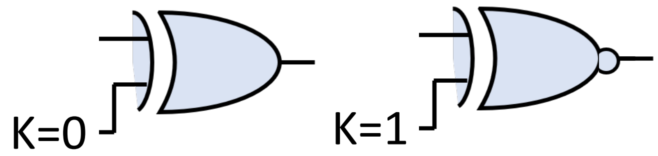
Broken oracle

- Analyze netlist to isolate key logic
- Infer the key from key logic or simply remove key logic to get IP

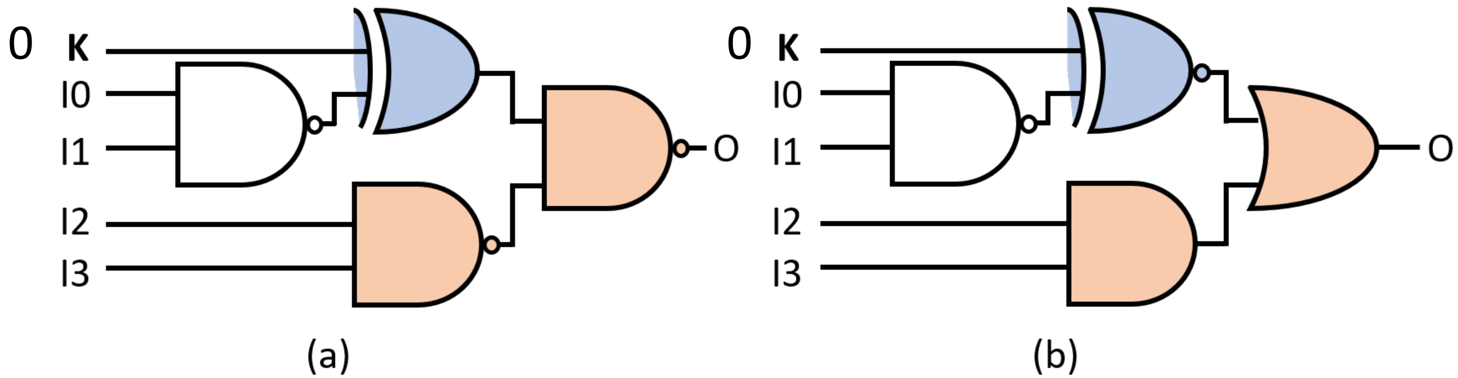
- **Likely attack angle**: Existing CAD tools are security-agnostic
  - They leave traces behind
  - They take predictable (learn-able) actions!

Traditional Logic Locking is Vulnerable

- Key-gate type **implies** the key value
  - ➢ Reverse-engineers can figure out the key!



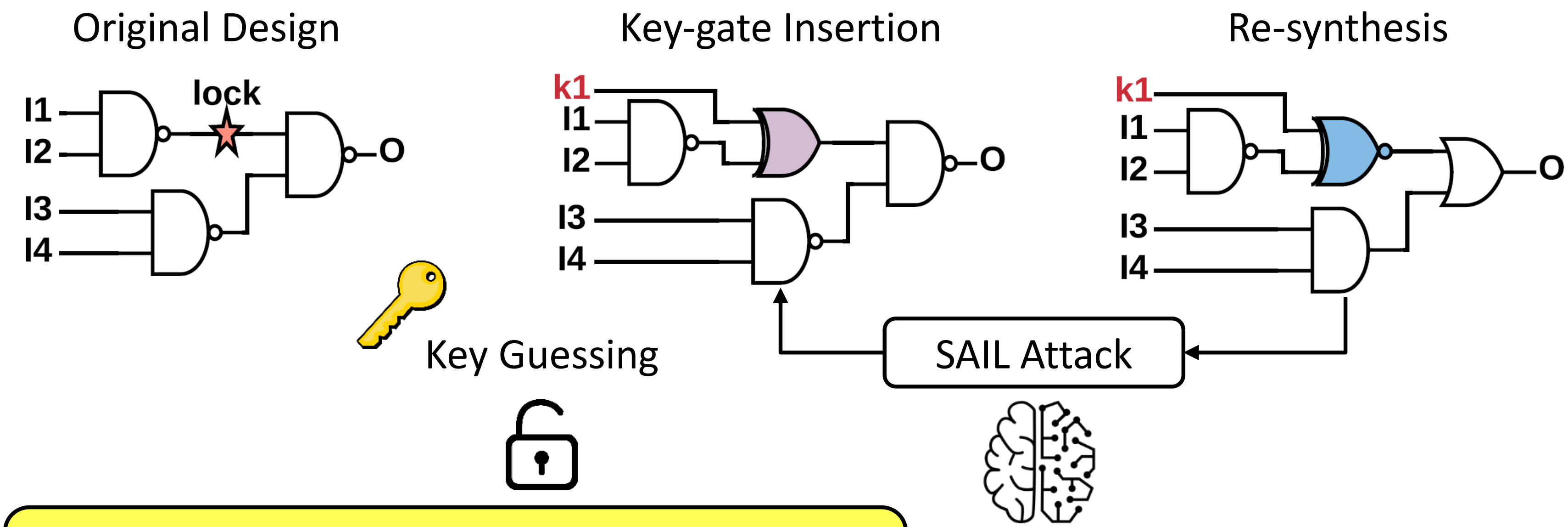
- Use bubble pushing feature of synthesis tools to **break** inference



- Attacks can **learn** the transformations and figure out the key (e.g., SAIL)
- **RLL is not really random**! It only chooses locations randomly.

SAIL Attack

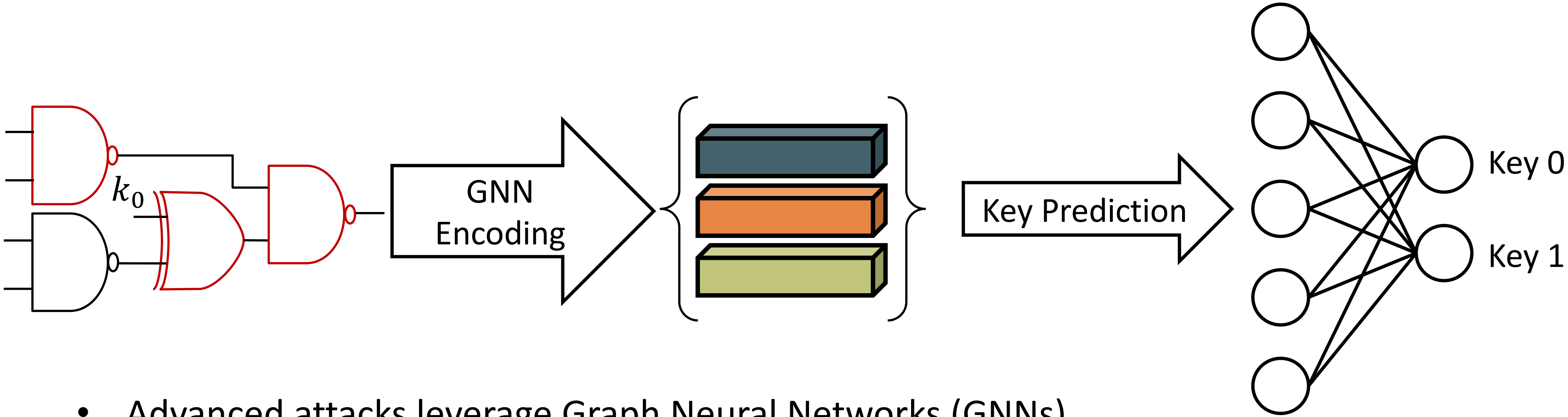
- Leverages the mapping between key-gate and key-value



- Logic locking induced changes are local
- Synthesis steps are deterministic

P. Chakraborty et al., “SAIL: Machine learning guided structural analysis attack on hardware obfuscation,” *AsianHOST*, 2018.

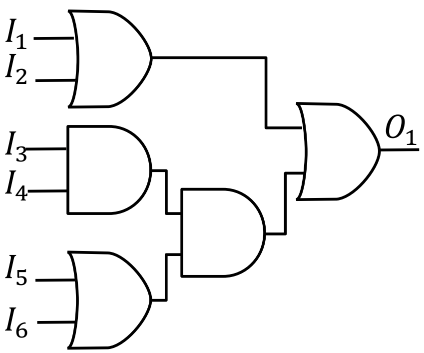
OMLA: Oracle-Less ML Attack on Logic Locking



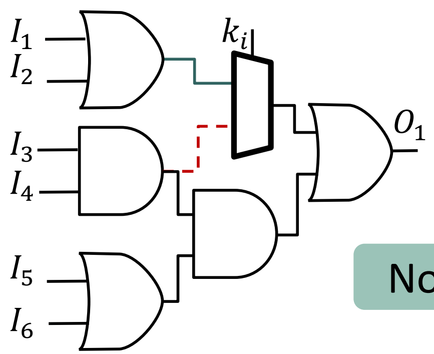
- Advanced attacks leverage Graph Neural Networks (GNNs)
- These attacks bypass the need to undo transformations
- GNNs can directly predict the key value from the key-gate locality

L. Alrahis, …, O. Sinanoglu, …, "OMLA: An Oracle-Less Machine Learning-Based Attack on Logic Locking," *TCAS-11*, 2021

Learning-resilient MUX-Based Logic Locking

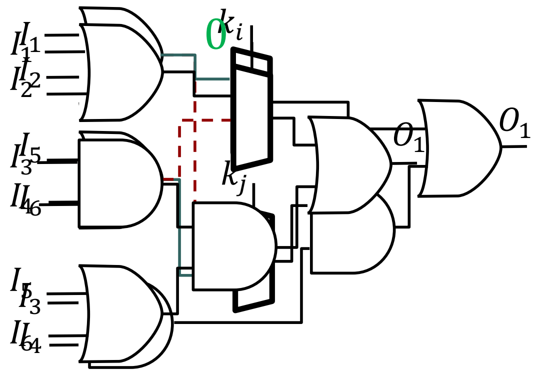


Original netlist

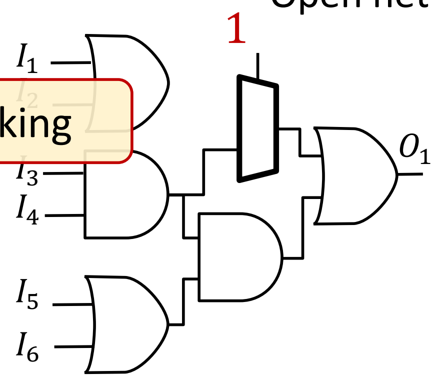


MUX locked netlist

No key leakage



Open net for $k_i = 1$



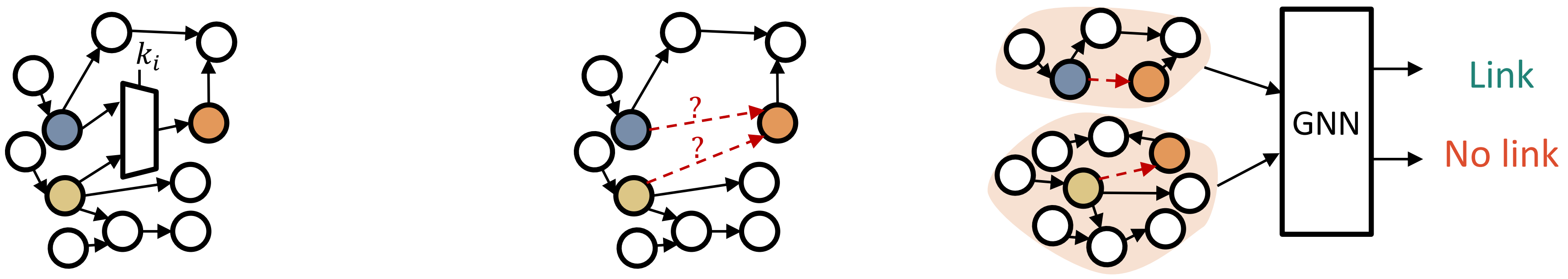
Solution? Symmetric/Deceptive MUX-based locking

Naïve MUX-based locking is vulnerable to constant propagation attacks!
E.g., SWEEP & SCOPE

A. Alaql *et al.,* "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *TVLSI*, 2021.
D. Sisejkovic *et al*., "Deceptive logic locking for hardware integrity protection against machine learning attacks," *TCAD,* 2021.

MuxLink on Learning-Resilient Logic Locking

- Modern ICs contain a large amount of repetition and reuse cores
- Symmetric/Deceptive logic locking only protects from locality-based attacks



MUX-based Locking

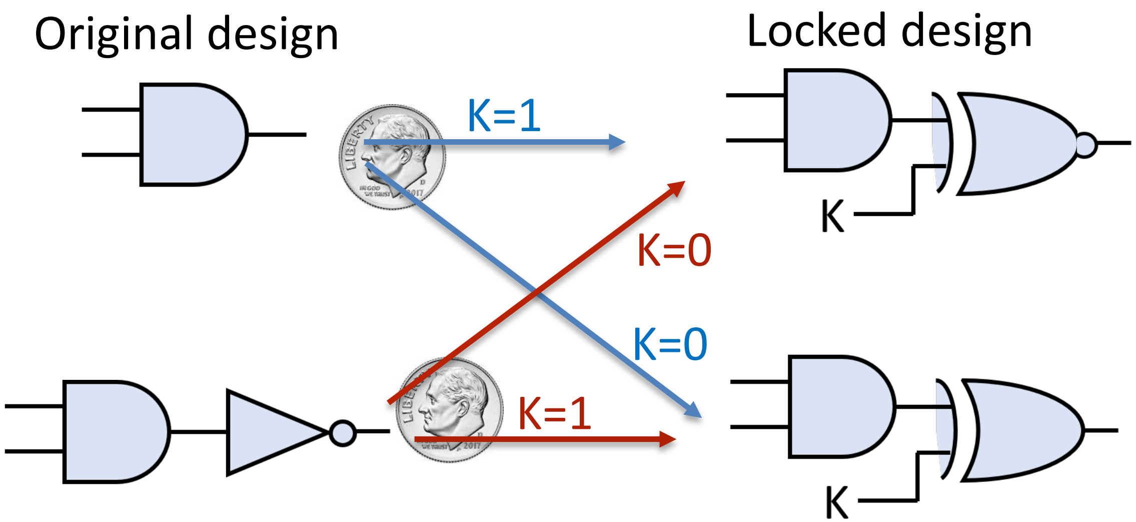
Converting to a link prediction task

Given an incomplete network, predict whether two nodes are likely to have a link

L. Alrahis, … O. Sinanoglu, "MuxLink: circumventing learning-resilient MUX-locking using graph neural network-based link prediction," *DATE, 2022.*

Truly Random Logic Locking

- Randomized decisions:
  - ➢ No inverter → insert XOR or XNOR
  - ➢ Inverter → replace with XOR or XNOR



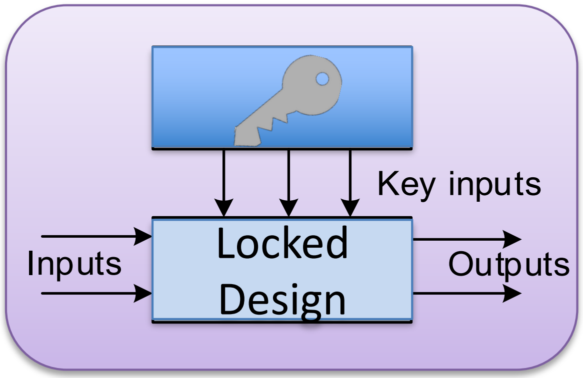
- No bubble pushing needed
- No reliance on logic synthesis tools

- **No inference of key value** from gate structure in locked design

N. Limaye, …, O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

Proposed Logic Locking

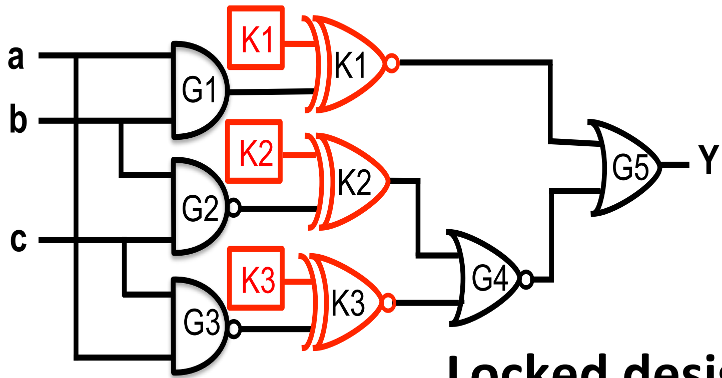
- Dishonest Oracle



When attack detected:
- **Withdraw** the key
- Oracle becomes **dishonest**
- Oracle-based attacks **fail**

- ML-Resilient Logic Locking

(Truly Random Logic Locking)



Locked design

Locking approach:
- **Randomized** decisions
- **No reliance** on synthesis tools
- High output **corruption**

N. Limaye, ..., O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

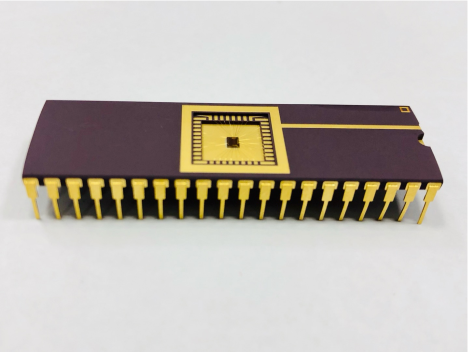
DISORC + TRLL Bulletproof?

- DISORC disables scan for oracle-based attacks
  - Attacks limited to chip I/Os not viable emprically
  - No provable security guarantees however

- TRLL provably securite against oracle-less attacks
  - Assumption: Attacker has zero knowledge about the original design

N. Limaye, …, O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

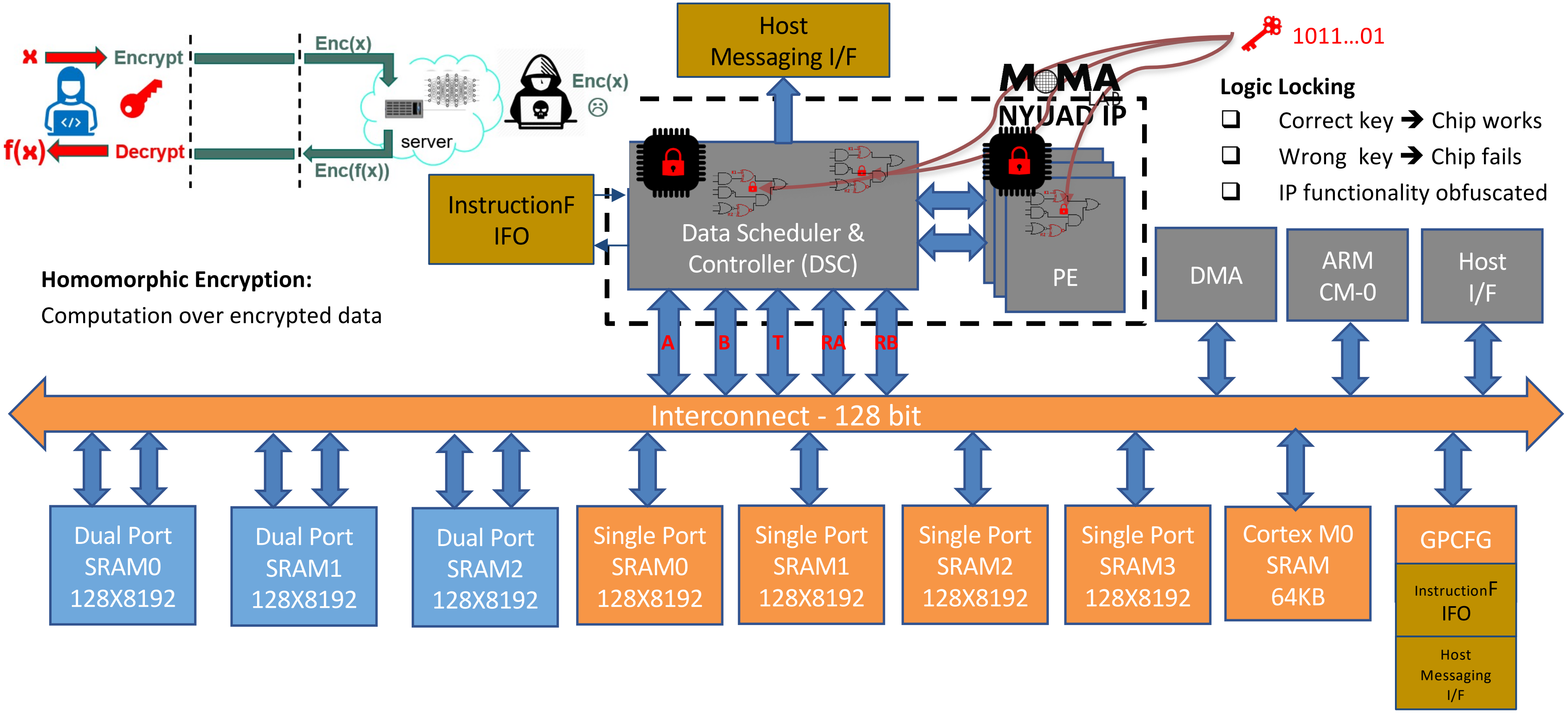
Logic-Locked Hardware Accelerator for Fully Homomorphic Encryption



جامعة نيويورك أبوظبي
NYU | ABU DHABI

CENTER FOR CYBER SECURITY

Chip Architecture



Summary

- Logic locking: A holistic defense
  - Regain control over supply chain (overbuilding, etc.)
  - Hide functionality (reverse engineering & IP piracy)
- Earlier research: Oracles force logic locking into an unpleasant trade-off on Error Rate
- DisORC + TRLL
  - Snaps this trade-off
  - Secure under certain assumptions

N. Limaye, ..., O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

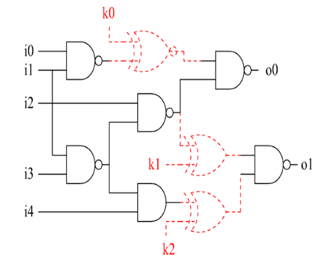
Future Direction: Security-Aware Logic Synthesis

Causal Nexus of “Logic Locking” & “Synthesis”

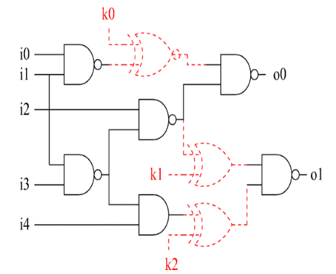
```
module example (/*AUTOARG*/
   // Outputs
   lower_out, o,
   // Inputs
   lower_inb, lower_ina, i
   );
   input i;
   output o;
   /*AUTOINPUT*/
   // Beginning of automatic inputs
   input  lower_ina; // To inst of inst.v
   input  lower_inb; // To inst of inst.v
   // End of automatics
   /*AUTOOUTPUT*/
   // Beginning of automatic output
   output lower_out; // From inst of inst.v
   // End of automatics
   /*AUTOREG*/
   // Beginning of automatic regs
   reg                  o;
   // End of automatics
   inst inst (/*AUTOINST*/
              // Outputs
              .lower_out (lower_out),
              // Inputs
              .lower_inb (lower_inb),
              .lower_ina (lower_ina));
   always @ (/*AUTOSENSE*/i) begin
      o = i;
   end
endmodule
```

Hardware design

Logic locking



State-of-the-art algorithms



RLL

Logic synthesis

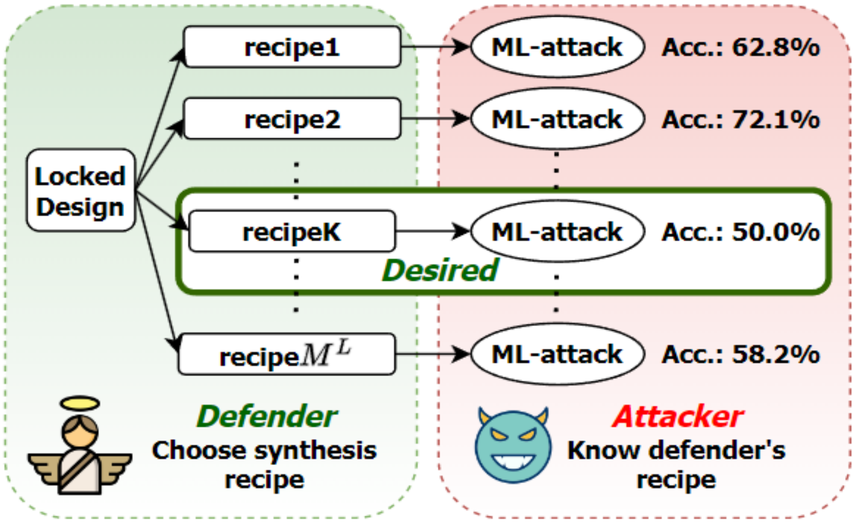
Vulnerable

Vulnerable

Long standing issue!

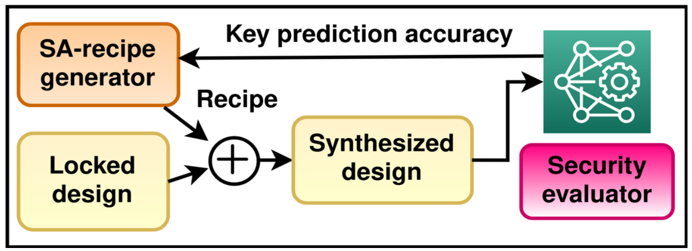
Future Direction: Security-Aware Logic Synthesis

Search space exploration



Synthesis **impacts** security

Solution



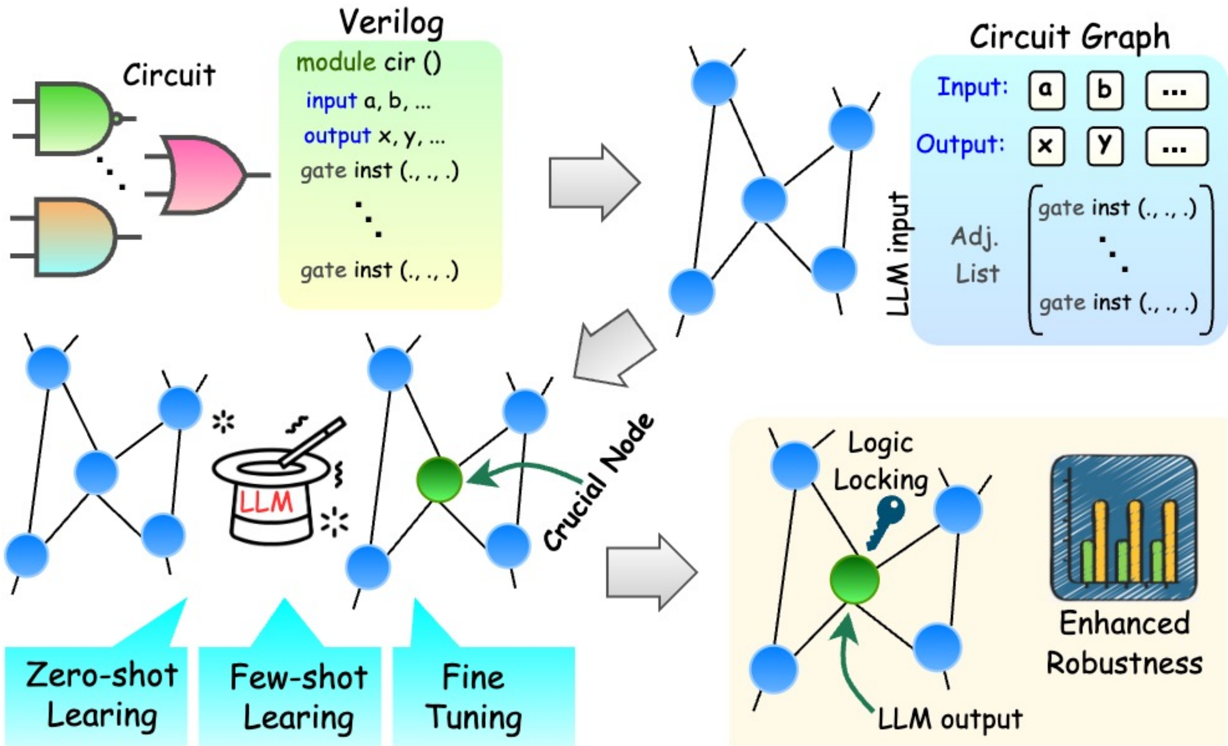
Optimization carried out using simulated annealing

Highlights potential for automation in logic locking

A. Chowdhury, L. Alrahis, O. Sinanoglu, ... "ALMOST: Adversarial learning to mitigate oracle-less ML attacks via synthesis tuning," *DAC, 2023*

Future Direction: Use of LLMs

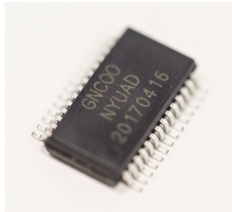
Can **LLMs** help identify *Influential nodes* in circuit graphs?

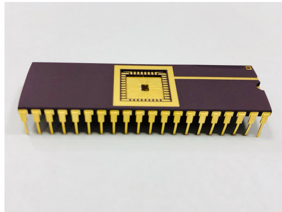


A. Saha, Sinanoglu, et al. IEEE VTS, 2025.

Useful Pointers

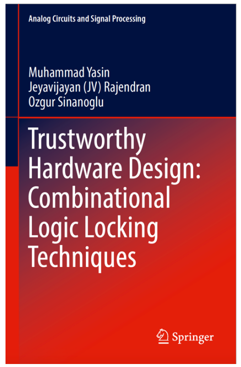
- Information on logic locking: https://sites.nyuad.nyu.edu/dfx/
  - Videos (lectures)
  - Material (for launching attacks)





- IP on logic locking
  - U.S. Patent No. 9,817,980.
  - U.S. Patent No. 10,153,769.
  - U.S. Patent No. 10,853,523.
  - U.S. Patent pending, US-20230177245-A1.

- Reference book



N. Limaye, ..., O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

Supply Chain Vulnerabilities of ICs and Mitigation Through Design-for-Trust

Ozgur Sinanoglu

جامعة نيويورك أبوظبي

NYU | ABU DHABI

21st IEEE SMACD Conference

July 10, 2025

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

NSF National Science Foundation WHERE DISCOVERIES BEGIN

TII

SRC® Semiconductor Research Corporation

ATIC Advanced Technology Investment Company A Mubadala Company

SCIENCE TO SHAPE THE FUTURE OF THE ARMY ARMY RESEARCH OFFICE 1951

GLOBAL FOUNDRIES

CENTER FOR CYBER SECURITY

Proposed Solution: Truly Random Logic Locking

- **<u>Objective 1</u>**: High corruptibility
  → Insert key-gates in random locations (like RLL)

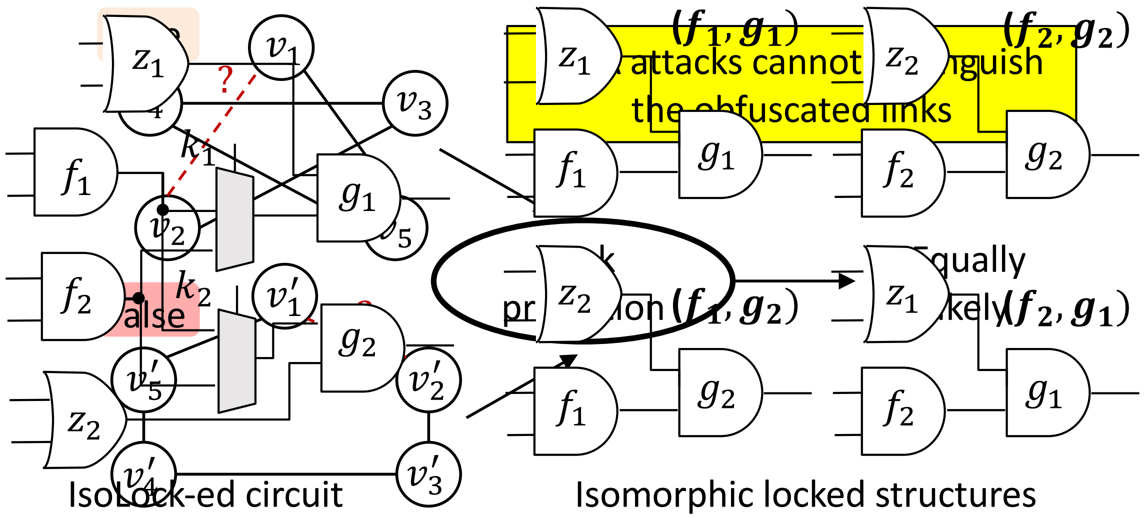
- **<u>Objective 2</u>**: Don’t rely on synthesis tools
  → Make randomized decisions
  → Eliminate inference between key-gate type and key value
  → Eliminate need for bubble pushing (TRLL)

N. Limaye, …, O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking," *IEEE TCAD*, 2021

Proposed Solution II: IsoLock

- MUX locking is inherently more secure (no key leakage)
- MUX locking is vulnerable to link prediction-based attacks
- What if we lock isomorphic structures?

Guaranteed resilience, but depends on the circuit structure

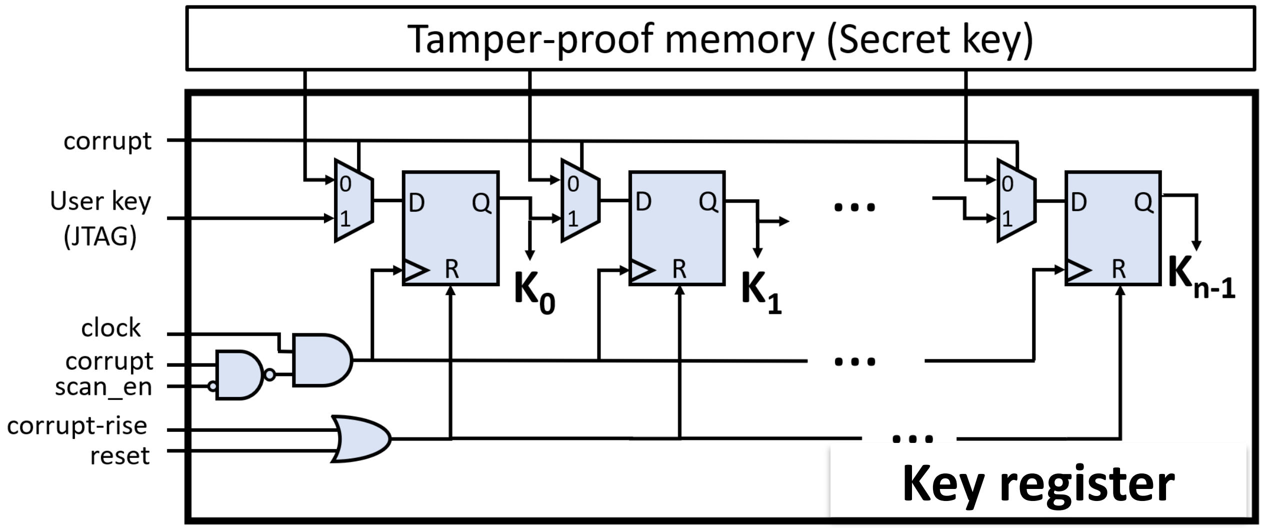


IsoLock-ed circuit

Isomorphic locked structures

L. Alrahis, ..., O. Sinanoglu, "IsoLock: Thwarting Link-Prediction Attacks on Routing Obfuscation by Graph Isomorphism," *Crypto Eprint Archive*

DisORC Implications on Test



Access to scan chains (scan-en = 1) means:

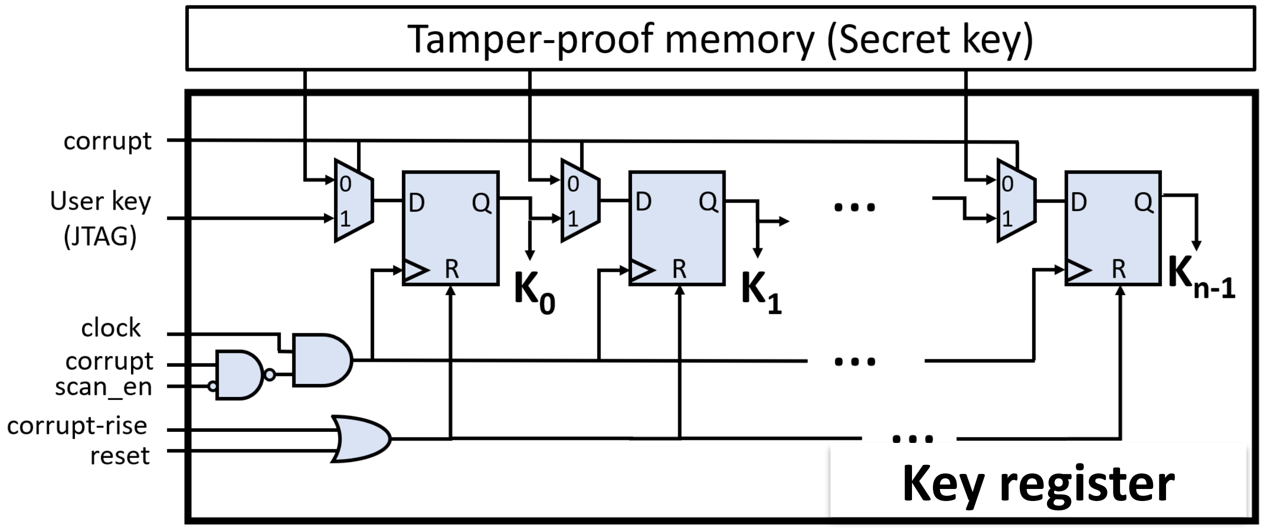
- **Attack** on logic locking
- **Legitimate testing** for structural defects

Does structural test require the correct functionality?

a. ~~Yes; need key in key register~~

b. No; can use key register as test points

DisORC Implications on Test (Cont'd)



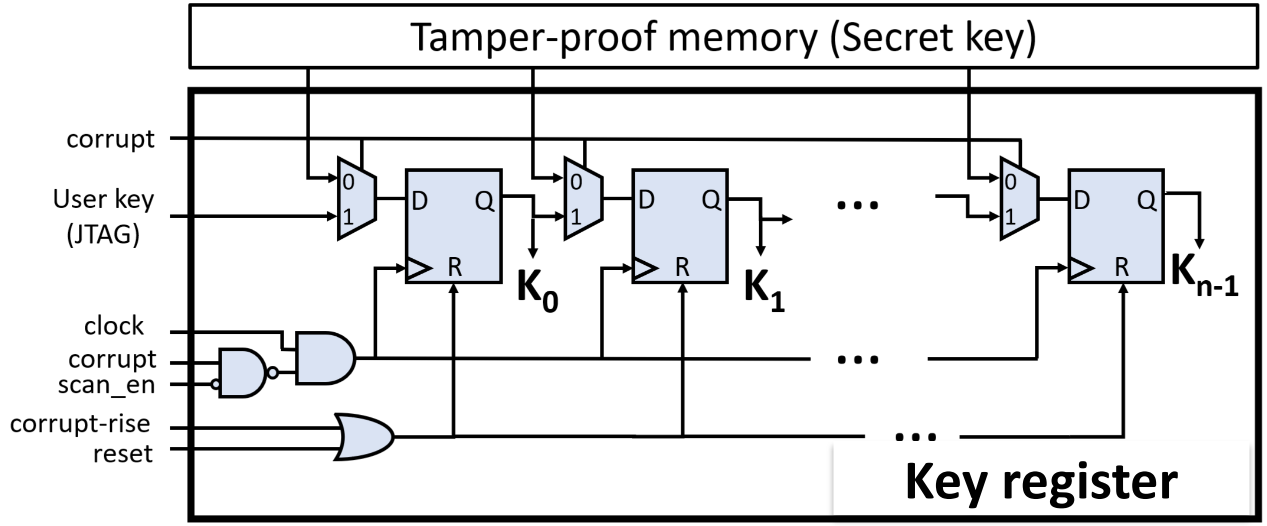
- ATPG sets key register content for each pattern
  - ➢ Fault coverage per pattern ↑
- Key register content loaded along with scan chains of the design

Does structural test require the correct functionality?

a. ~~Yes; need key in key register~~

b. No; can use key register as test points

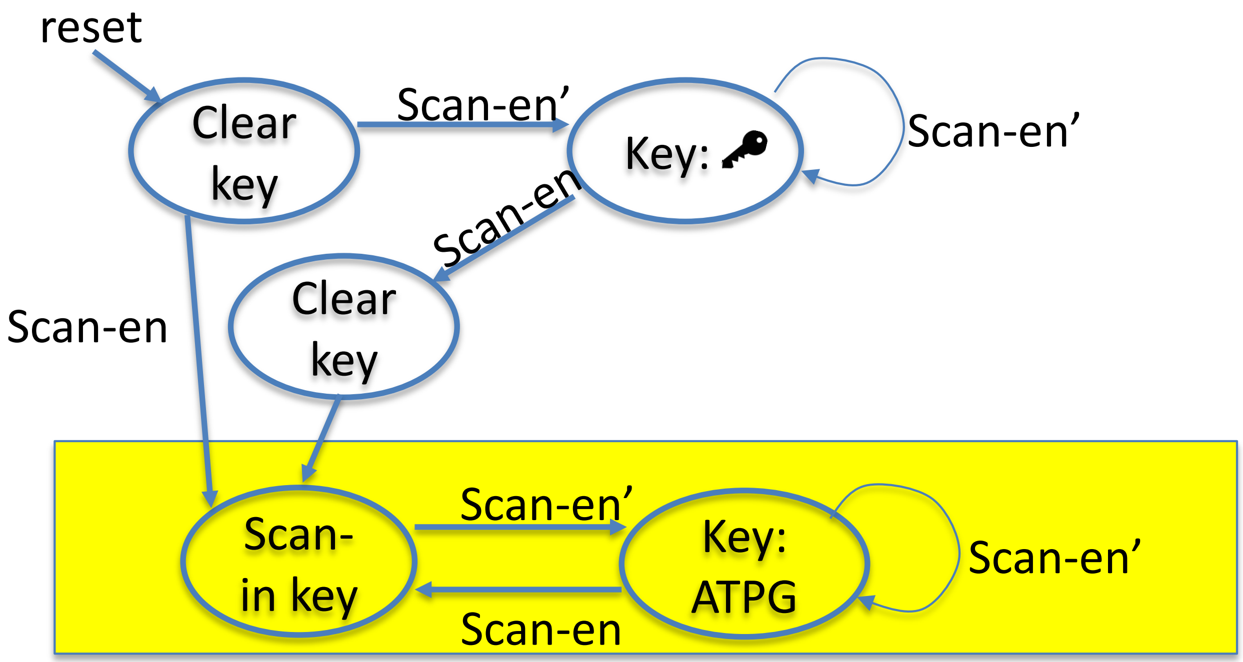
DisORC Implications on Test (Cont'd)



- ATPG sets key register content for each pattern
  - ➢ Fault coverage per pattern ↑
- Key register content loaded along with scan chains of the design

- **Key** isolated and hidden during test
- No info in test patterns about **key**
- Chips with **key** can be tested by **untrusted** OSAT

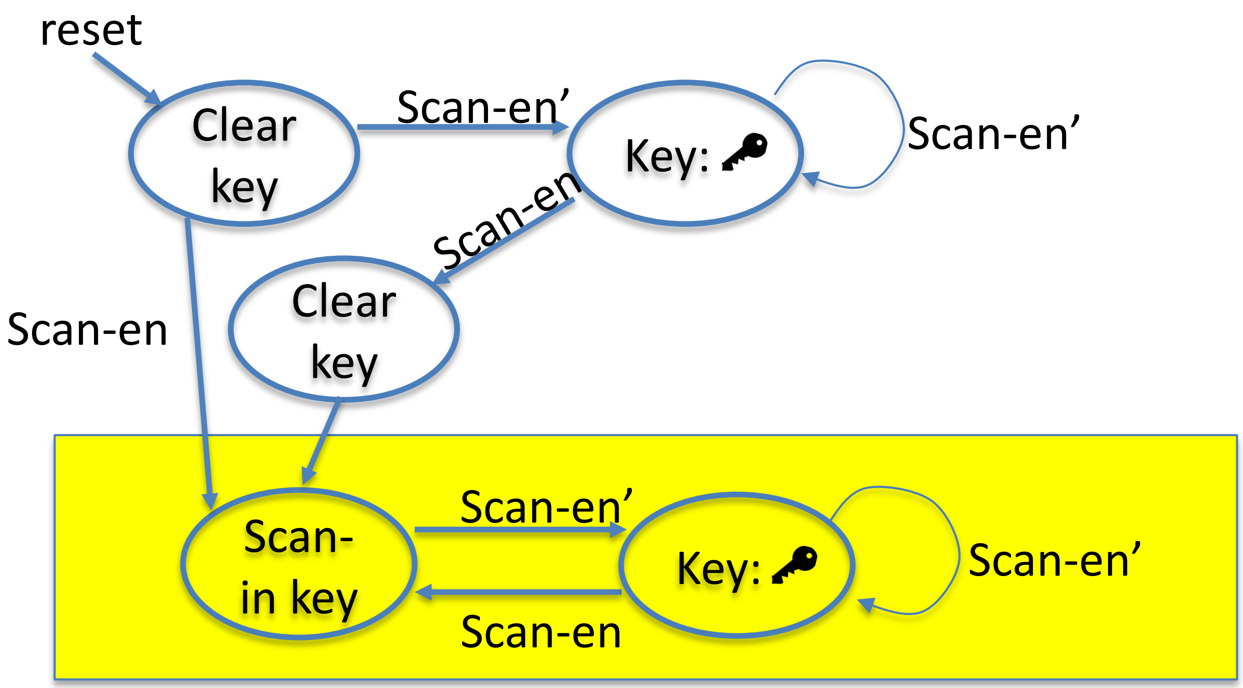
DisORC Implications on Test (Cont'd)



- ATPG sets key register content for each pattern
  - Fault coverage per pattern ↑
- Key register content loaded along with scan chains of the design

- **Key** isolated and hidden during test
- No info in test patterns about **key**
- Chips with **key** can be tested by **untrusted** OSAT

DisORC Implications on Debug



Access to scan chains (scan-en = 1) means:

- **Attack** on logic locking
- **Legitimate debug** of mission mode failures

Debug must be done in a **trusted facility**

- Scan-in: **Secret key** loaded from JTAG along with initial scan state
- Functional mode
- Scan-out

Attacks Vs Defenses

Attack \ Defense	Oracle access	RLL [14]	FLL [15]	SLL [16]	SARLock [24]	Anti-SAT [17]	SFLL-HD [18]	SFLL-fault [19]
~~Sensitization [16]~~	Yes	✗	✗	✓	✓	✓	✓	✓
~~SAT [20]~~	Yes	✗	✗	✗	✓	✓	✓	✓
~~AppSAT [21]~~	Yes	✗	✗	✗	✓	✓	✓	✓
~~Double DIP [22]~~	Yes	✗	✗	✗	✓	✓	✓	✓
~~Bypass [23]~~	Yes	✓	✓	✓	✗	✗	✓	✓
~~SPS [26]~~	No	✓	✓	✓	✗	✗	✓	✓
~~AGR [26]~~	Yes	✓	✓	✓	✗	✗	✓	✓
Redundancy [29]	No	✗	✗	✗	✓	✓	✓	✓
~~FALL [28]~~	No	✓	✓	✓	✓	✓	✗	✓
SAIL [30]	No	✗	✗	✗	✓	✓	✓	✓
~~Approximate use of circuit~~	Yes	✓	✓	✓	✗	✗	✗	✗

✓- Defense is resilient to the attack ✗- Defense is vulnerable to the attack

- DisORC thwarts oracle-guided attacks
- Can now use a **high corruptibility** logic locking scheme in tandem!
- Now focus on oracle-less attacks (don't rely on synthesis tools!)

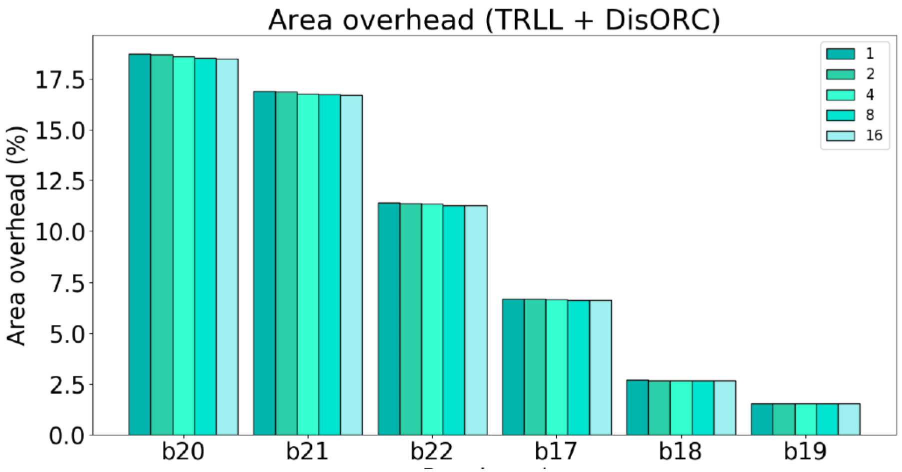
Results: Netlist-Analysis-Based Attacks

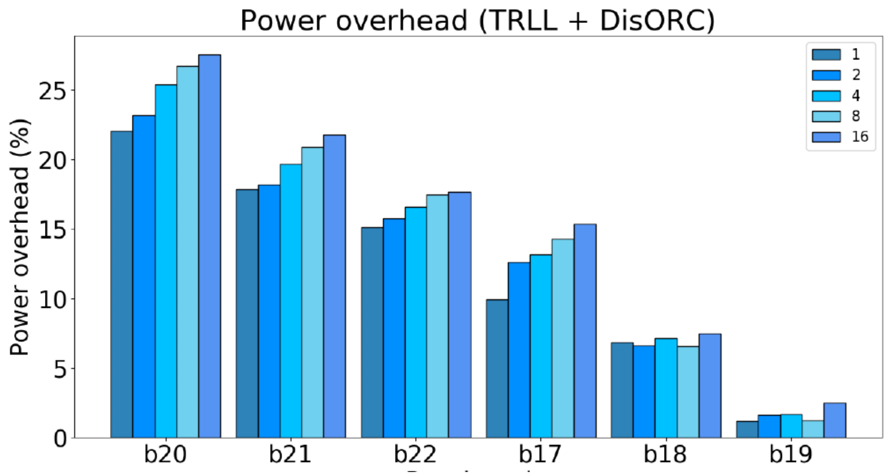
Benchmark	Key-size	Recovered key (%)	Execution time (hr)
b20	236	34.74	3.44
b21	229	38.86	2.76
b22	243	43.62	7.93
b17	256	38.28	21.38
b18	97	-	-
b19	208	-	-

- Redundancy attack* applied on **DisORC+TRLL defense**
  - Did not terminate on the largest circuits
  - Recovered <50% of the key (~random-guess) on small circuits

*Li & Orailoglu, DATE 2019

Results: Implementation Cost





Area and power overhead (for iso-performance) for 1, 2, 4, 8, 16 scan chains
DisORC + TRLL with 128-bit logic locking
Largest circuit: **b19**; 65K gates, 6.5K flip-flops

- Cost gets smaller for larger circuits
- b19: 1.5% area, 1.2% power

Results: Test Cost & Quality

Circuits	Fault coverage (%)		# Test patterns		Test data volume (bits)	
	Original	Locked	Original	Locked	Original	Locked
b20	99.99	100	411	415	425K	482K
b21	100	100	439	373	453K	433K
b22	100	100	408	416	595K	660K
b17	99.95	99.99	533	467	1,578K	1,443K
b18	100	99.99	623	617	4,111K	4,150K
b19	99.82	99.82	836	858	10,980K	11,379K

Test data volume = # Test patterns x # bits per pattern

- Fault coverage same
- Slight increase in test data volume (due to test points)
  - b19: 3.6%

DisORC + TRLL = Bullet-Proof Logic Locking

Attack \ Defense	Oracle access	RLL [14]	FLL [15]	SLL [16]	SARLock [24]	Anti-SAT [17]	SFLL-HD [18]	SFLL-fault [19]	DisORC+TRLL
Sensitization [16]	Yes	✗	✗	✓	✓	✓	✓	✓	✓
SAT [20]	Yes	✗	✗	✗	✓	✓	✓	✓	✓
AppSAT [21]	Yes	✗	✗	✗	✓	✓	✓	✓	✓
Double-DIP [22]	Yes	✗	✗	✗	✓	✓	✓	✓	✓
Bypass [23]	Yes	✓	✓	✓	✗	✗	✓	✓	✓
SPS [26]	No	✓	✓	✓	✗	✗	✓	✓	✓
AGR [26]	Yes	✓	✓	✓	✗	✗	✓	✓	✓
Redundancy [29]	No	✗	✗	✗	✓	✓	✓	✓	✓
FALL [28]	No	✓	✓	✓	✓	✓	✗	✓	✓
SAIL [30]	No	✗	✗	✗	✓	✓	✓	✓	✓
Approximate use of circuit	Yes	✓	✓	✓	✗	✗	✗	✗	✓

✓- Defense is resilient to the attack ✗- Defense is vulnerable to the attack

IEEE TCAD
10.1109/TCAD.2020.3029133

Protection from:

- Foundry, OSAT, and end-users (or all of them colluding)

Attacks Vs Defenses

Attack \ Defense	Oracle access	RLL [14]	FLL [15]	SLL [16]	SARLock [24]	Anti-SAT [17]	SFLL-HD [18]	SFLL-fault [19]
Sensitization [16]	Yes	✗	✗	✓	✓	✓	✓	✓
SAT [20]	Yes	✗	✗	✗	✓	✓	✓	✓
AppSAT [21]	Yes	✗	✗	✗	✓	✓	✓	✓
Double-DIP [22]	Yes	✗	✗	✗	✓	✓	✓	✓
Bypass [23]	Yes	✓	✓	✓	✗	✗	✓	✓
SPS [26]	No	✓	✓	✓	✗	✗	✓	✓
AGR [26]	Yes	✓	✓	✓	✗	✗	✓	✓
Redundancy [29]	No	✗	✗	✗	✓	✓	✓	✓
FALL [28]	No	✓	✓	✓	✓	✓	✗	✓
SAIL [30]	No	✗	✗	✗	✓	✓	✓	✓
Approximate use of circuit	Yes	✓	✓	✓	✗	✗	✗	✗

✓- Defense is resilient to the attack ✗- Defense is vulnerable to the attack

Highly effective but ~~**oracle-vulnerable**~~ + Oracle-resilient but **barely effective** ?