

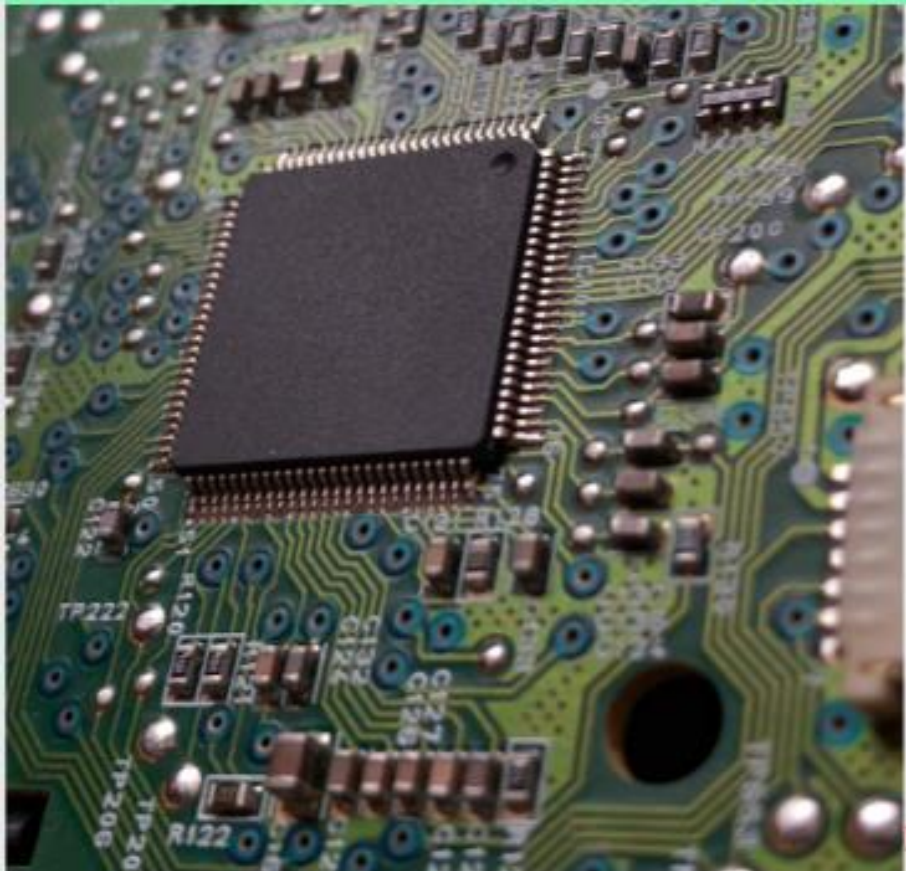


ProbLock

BY MARGARITIS PASQUALE AND NIKOU RAPHAEL

ELECTRICAL AND COMPUTER ENGINEERING- University of Western Macedonia

Table of Contents



01	Summary	05	Experimental Results
02	Historical Development	06	Discussion
03	Related Work	07	Coclusions and Future Work
04	ProbLock	08	Questions

01.

Summary



Security attacks



Integrated circuit (IC) piracy
and overproduction

Threaten the
security and integrity of a
system





Logic locking is a type of hardware obfuscation technique where additional key gates are inserted into the circuit.

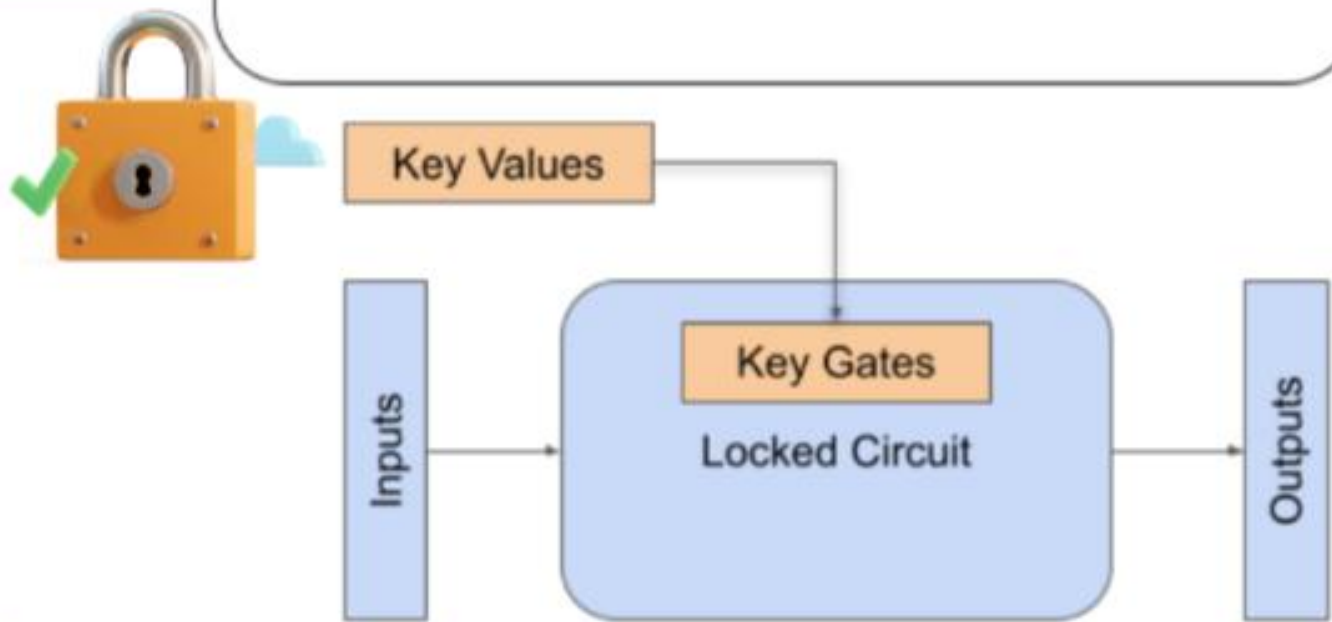
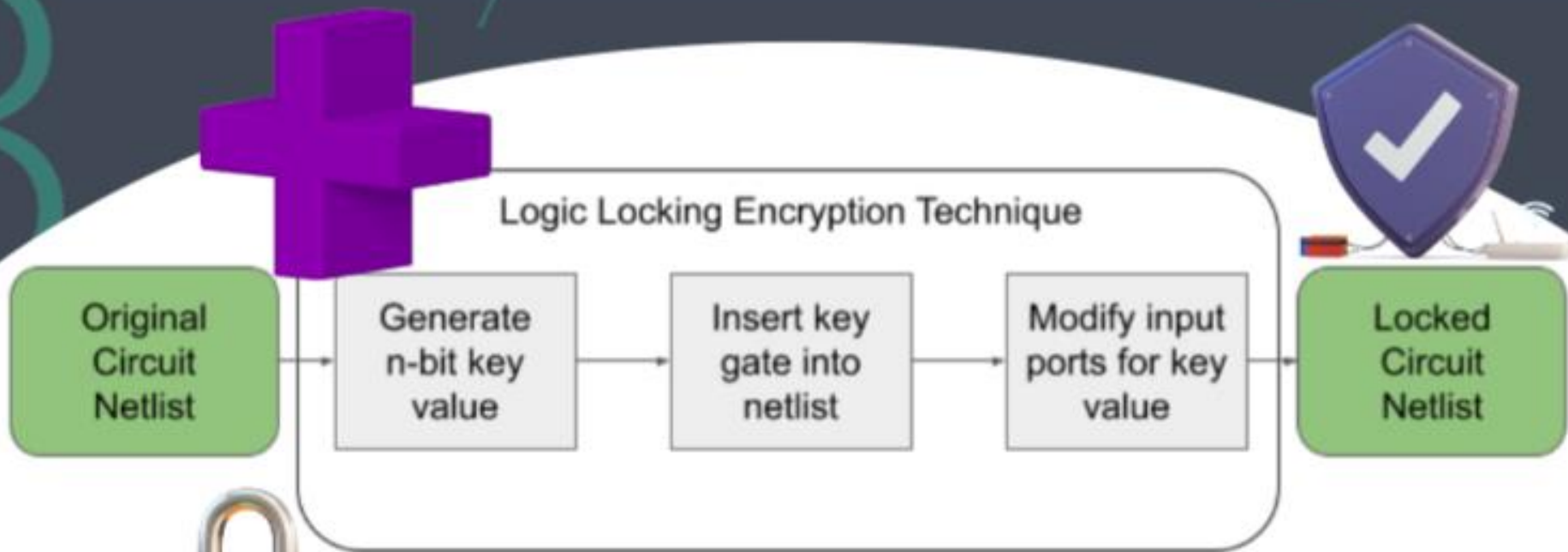


Only the correct key can unlock the functionality of that circuit; elsewhere it provides wrong output.

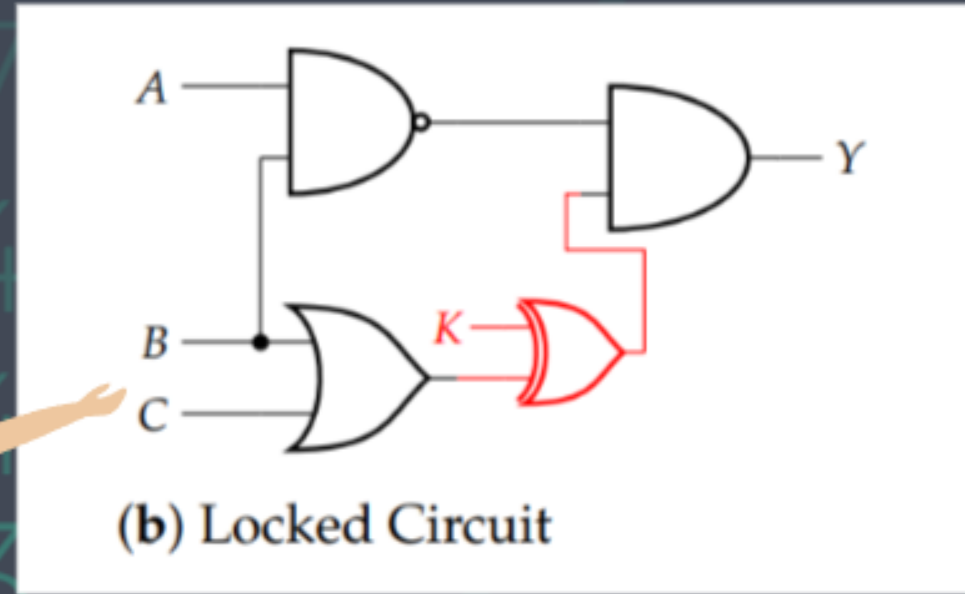
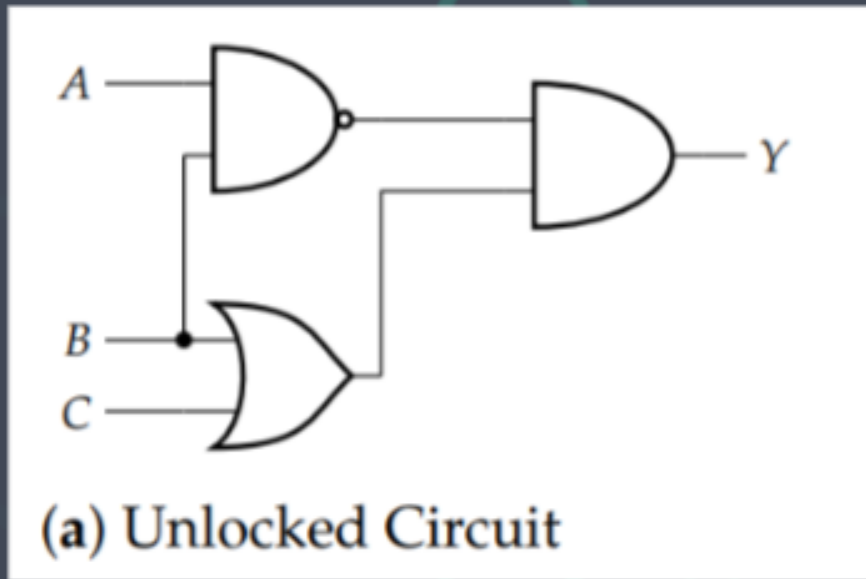


The article proposes ProBlock, which can be used to both:

- Combinational
- Sequential circuits
- with a critical selection process



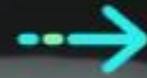
Using XOR and XNOR components as key gates, the proper key value will make the gate act as a buffer and have no effect on the rest of the logic. If the wrong key value is provided, the key gate will produce a wrong value and make the circuit nonfunctional.



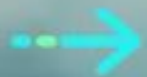
02.

Historical Development



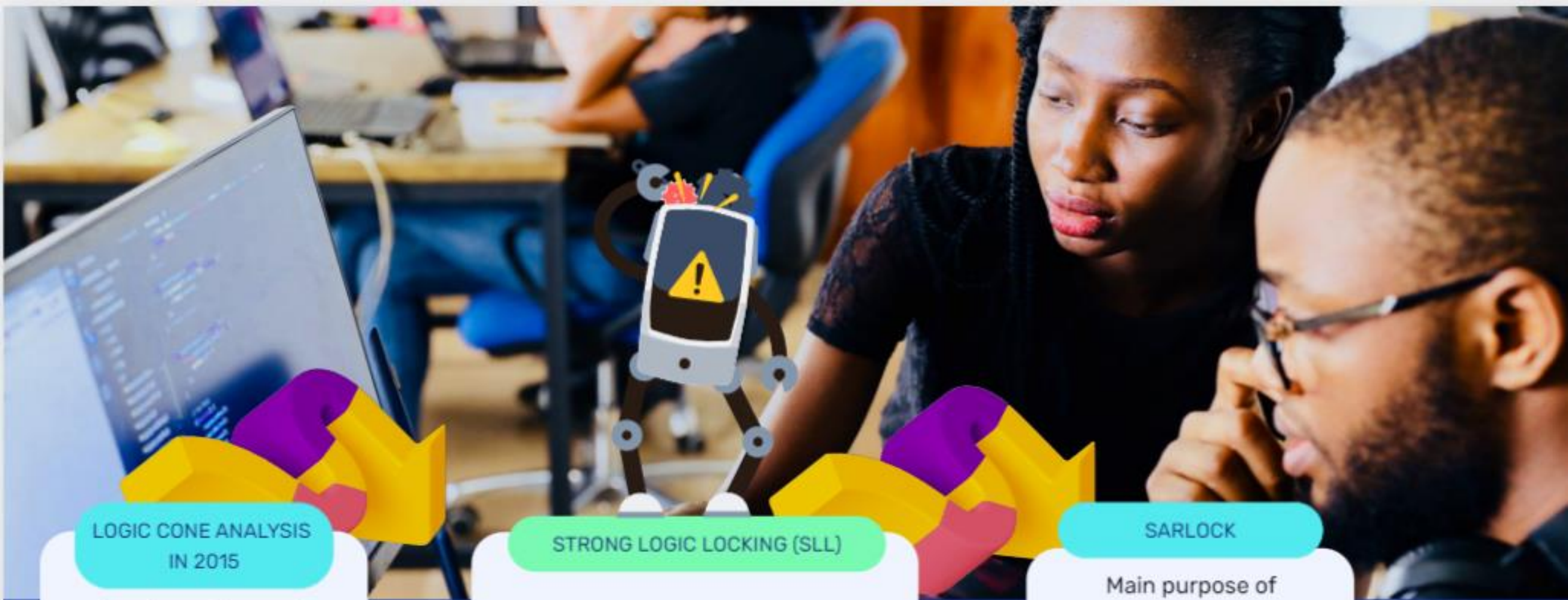


SAT attacks probed the input and output patterns of the system to determine the key to unlock a circuit.



SAT attacks have proven to be very effective against logic locking techniques.





LOGIC CONE ANALYSIS IN 2015

Fan-in and Fan-out
metrics to insert key
gates
into a netlist

STRONG LOGIC LOCKING (SLL)

Analyzed the relationship
between inserted key gates
in the form of a graph

SARLOCK

Main purpose of
thwarting SAT
attacks. The
method made SAT
attacks exponential in
complexity and
therefore ineffective.



**CRITICAL
ERROR**

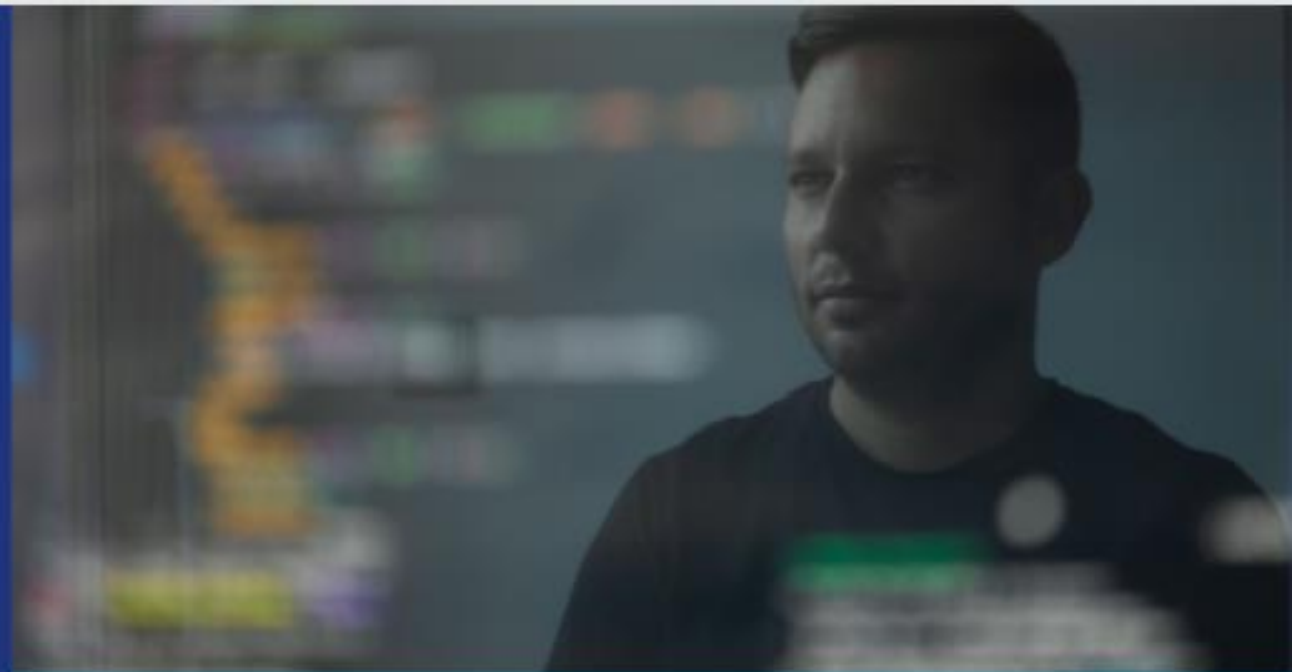


Vulnerable to sensitization exploits and strong Oracle-based attacks



03.

Related Work



FIRST TECHNIQUES

One of the earliest logic-locking techniques inserted key gates randomly into the circuit.

STRONG LOGIC LOCKING (SLL)

SLL was initially evaluated with a hill-climbing attack where the bits of an initial random key guess is toggled to minimize hamming distance between circuit outputs and test responses. If a key produces a hamming distance of 0, the attack is considered successful.

Ineffective at determining the correct key value for all tested ISCAS '85 benchmarks



Basis of the attack model to determine where to insert key gates



SARLock employs a small overhead strategy that exponentially increases the number of distinguishing input patterns (DIPs) needed to unlock the circuit





Calculating the number of DIPs needed to determine the correct key value to unlock a locked circuit.

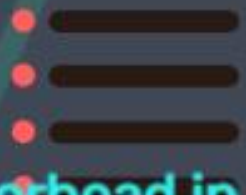


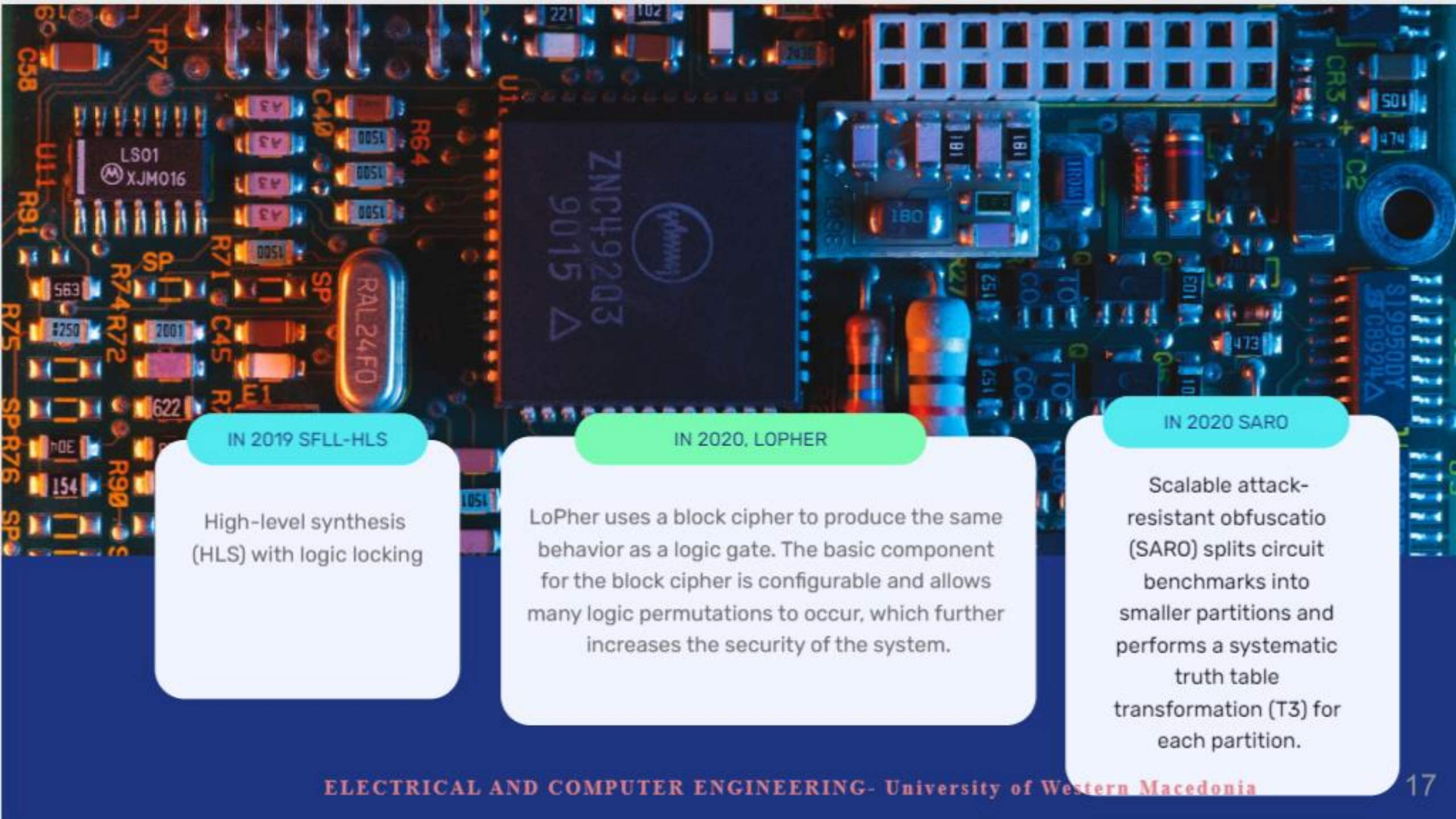
SarLock proved that it was more effective against SAT attacks because it took a larger number of DIPs and more time to break the circuit



The SAT algorithm would run for hours to break a SarLock circuit, but it took less than a second for all SLL circuits.

- In 2017, TTLock was proposed, which resisted all known attacks including SAT and sensitization attacks
- TTLock would invert the response to a logic cone to protect the input pattern
- The logic cone would be restored only if the correct key is provided.
- Stripped-functionality logic locking (SFLL)
- Did not account for the cost of tamper-proof memory, lead to high overhead in the re-synthesis process





IN 2019 SFLL-HLS

High-level synthesis (HLS) with logic locking

IN 2020, LOPHER

LoPher uses a block cipher to produce the same behavior as a logic gate. The basic component for the block cipher is configurable and allows many logic permutations to occur, which further increases the security of the system.

IN 2020 SARO

Scalable attack-resistant obfuscation (SARO) splits circuit benchmarks into smaller partitions and performs a systematic truth table transformation (T3) for each partition.

04.

ProbLock





ProbLock is a probability-based technique that inserts key gates into a circuit netlist where only the correct key value will unlock the circuit to narrow down a set of best nodes to insert key gates.



A Probability constraint is the main metric that they used to lock the circuits.



Four constraints to determine the best candidate nodes to insert XOR or XNOR key gate:

- Longest path
- Non-critical path
- Low dependent nodes
- Best probability nodes



The first three constraints find the set of nodes that lie on the longest path and non-critical path and have low-dependence wires.



The last constraint uses probability to find the set of nodes equal to the key length where the key gates will be inserted.

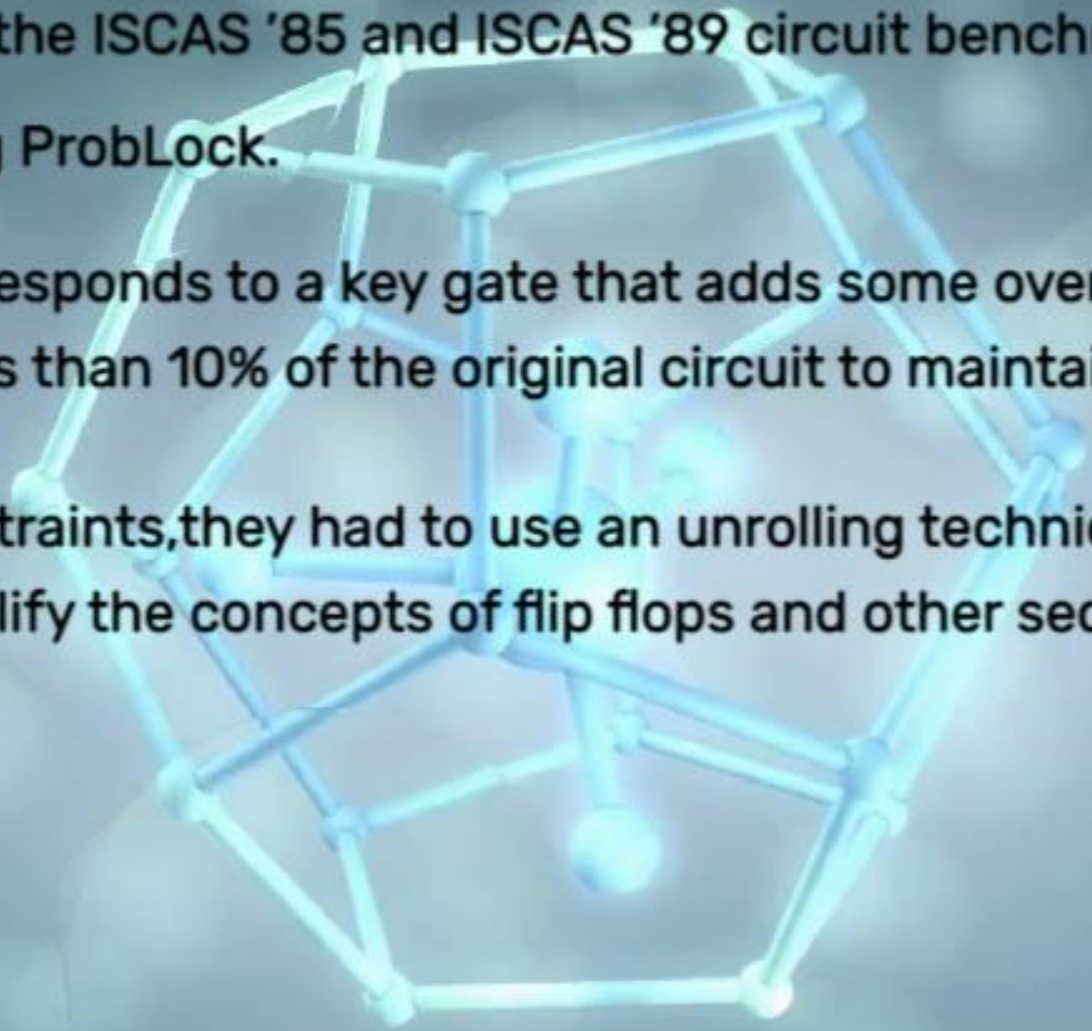


The longest path and non-critical path is chosen:

- to avoid critical timing elements
- and to insert key gates on parts of the circuit that were being used the most

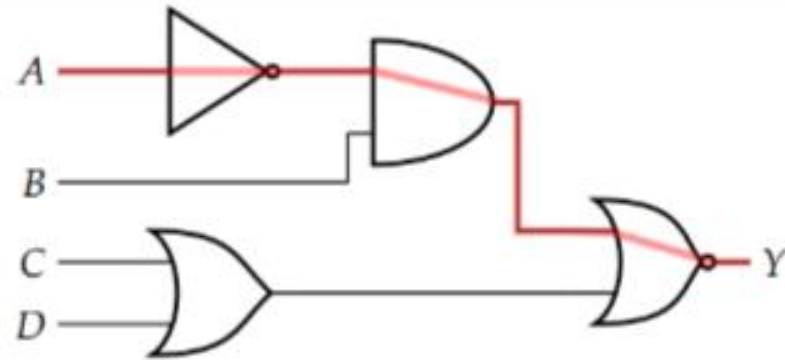
The research model

- For the tests of the research, there were used a set of combinational and sequential circuit netlists using the ISCAS '85 and ISCAS '89 circuit benchmarks.
- 40 benchmarks using ProbLock.
- Each bit of a key corresponds to a key gate that adds some overhead to the circuit so that the key size has to be less than 10% of the original circuit to maintain a low overhead.
- For some of the constraints, they had to use an unrolling technique to accurately filter out nodes (simplify the concepts of flip flops and other sequential logics)

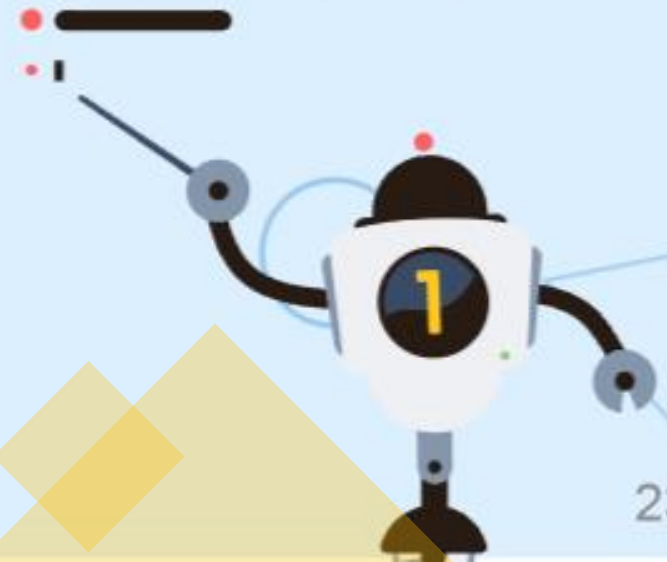


The longest path constraint

- The longest path constraint isolates a subset of nodes that lie on the longest paths in a circuit netlist.
- They represent the netlist of each benchmark as a directed acyclic graph (DAG)
- Each vertex in the DAG is a gate element from the netlist, and each vector represents the wire connecting to the next gate element
- Once the DAG is constructed for each benchmark, they calculated the longest paths of the DAG using a depth first search (DFS) technique.

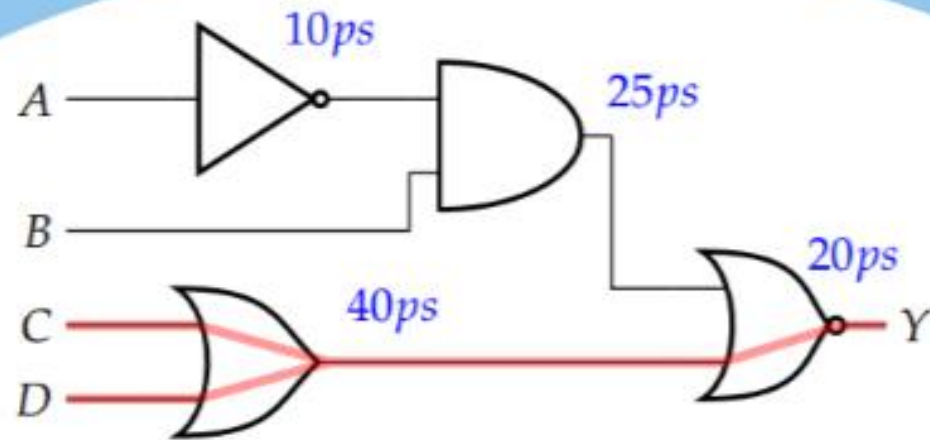


Longest Path (in red).



The critical path constraint

- Similar to the Longest path constraint, however rather than considering logic depth, they look at timing information.
- This constraint is essential, as adding gates on the critical path could break the circuit functionality or change timing specifications.



Critical paths (in red).

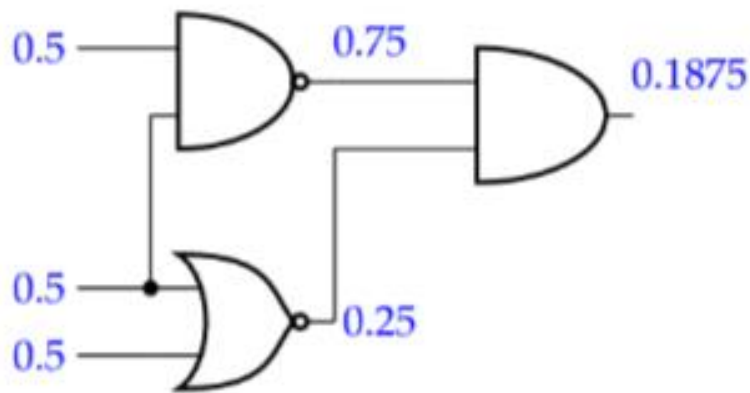
The low-dependence constraint

- The output wire of a gate is considered low dependence if the input wires to that gate have little influence on the value of output.(FANSI)
- A functional truth table is created for each output wire of each gate in the circuit.
- The value for each input gets stored as a list for each output wire.
- Low-dependent wires are weak spots in the circuit so this constraint isolates those locations
 - in order to improve security.
- We insert key gates next to low-dependence wires to fortify any weaknesses.The filtering process passes the subset of nodes to the final constraint.

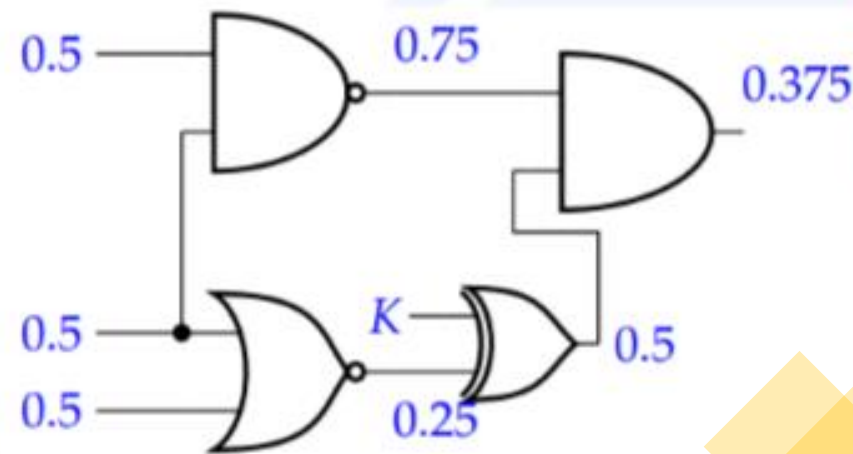


Probability constraint

- The probability constraint focuses on reducing the effectiveness of the SAT attacks.
- In a SAT attack, a distinguishing input (DI) is chosen and the attacker runs through various key values, eliminating any that yield an incorrect output.
- To reduce the effectiveness of a SAT attack, the number of wrong keys produced for a given DI must decrease. (This can be done by bringing the probability of any given node being 1 closer to 0.5)



(a) Pre-Insertion Probabilities



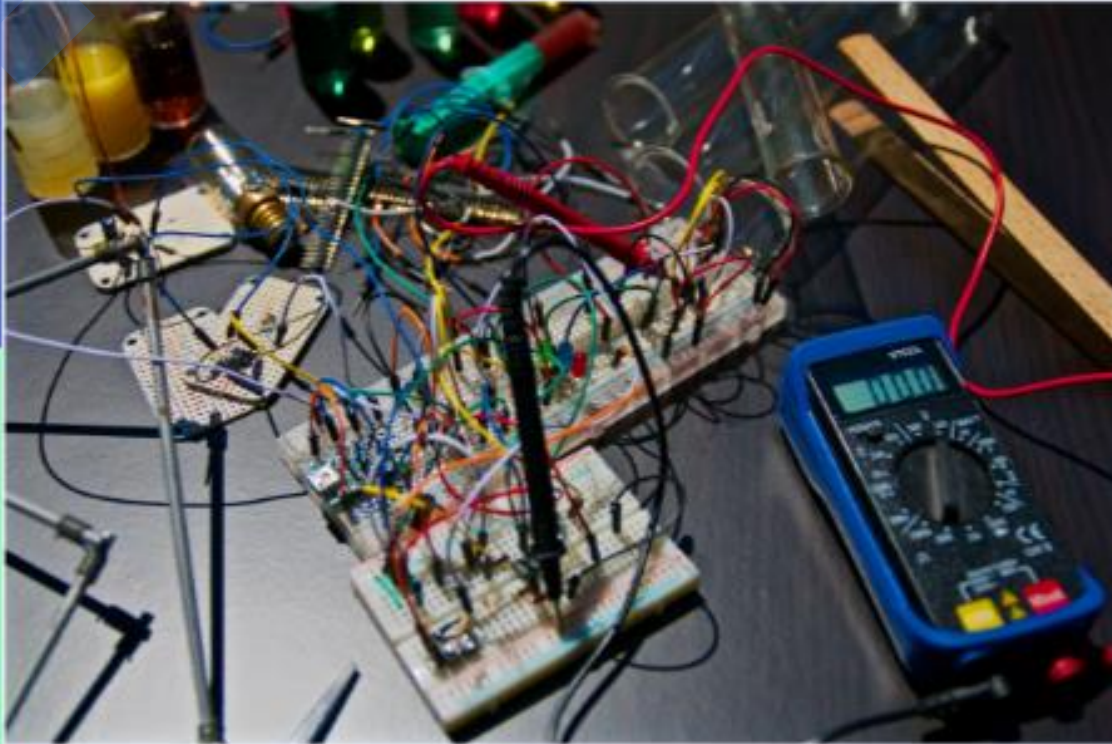
(b) Post-insertion probabilities



05.

Experimental Results





They analyzed the correlation and relationship between the constraints



They chose two constraints based on path elements and two constraints based on wires and nodes.



The strength of the correlation varies between benchmarks because of the shape and functionality of each circuit.



Enough nodes were removed with each subset until final set of best candidates was discovered.



For the final biased probabilities constraint, the final of nodes was equal to the size of the key.



After re-synthesizing the obfuscated netlists, they used Synopsys to verify the behavior of the locked circuits



The critical path of the netlists remained the same, and the timing analysis remained consistent for all benchmarks

They tested ProbLock against other encryption techniques from similar logic-locking papers. (Toc13xor and dac12)

The results show that ProbLock is more secure than the other techniques for the majority of the combinational benchmarks.

Each technique that they used to lock a circuit, including ProbLock, had less than 10% area overhead.

06.

Discussion





For most of the combinational benchmarks, ProbLock has a longer decryption time and better security. (probability constraint presented by ProbLock is a novel metric).



They only tested combinational circuits



PROs of the technique:

- simplicity of the algorithm(. Only one additional logic gate is required for each bit of the key to lock a circuit)
- very small overhead
- while gaining maximum security

Negatives of ProbLock

- Some of the limitations of ProbLock include the high complexity and lack of experimental data needed for comparison.
 - The algorithm for the probability constraint is performed k times for each bit of the key.
- This adds a large amount of computation time that can be improved and optimized in later efforts.
- Most of the algorithms used to support ProbLock can also be optimized for a lower algorithmic complexity.



07.

Conclusions and Future Work





In the future, they intend to test the obfuscated benchmarks against known attacks and compare them to other logic locking techniques (as SLL, logic cone locking, and SARLock)



They also plan on strengthening ProbLock against SAT attacks by integrating SAT-resistant logic near the key gate locations. This would increase overhead, but they would also try to optimize this in their experiment.



They will use ProbLock in order to:

- building an anti-SAT block
- exponentially increase the number of iterations in the SAT attack algorithm.

08.

Any Questions?





THANK
YOU