Physically Secure Hardware Redaction and Logic Locking with Hybrid Logic Systems

Alexander N. Chin*, Jared D. Arzate*, Yiorgos Makris,
Naimul Hassan, Alexander J. Edwards, Joseph S. Friedman
Electrical and Computer Engineering, The University of Texas at Dallas
Richardson, TX 75080; {alexander.edwards, joseph.friedman}@utdallas.edu

***Abstract*—While hardware obfuscation techniques like logic locking and hardware redaction can secure chips against algorithmic attacks, security against physical attacks requires the use of emerging technologies with large error rates. Hybrid logic systems have therefore been proposed in which polymorphic islands with physical security provide algorithmic security within a large CMOS system. In this work, we demonstrate that a well-chosen encryption of small, physically secure islands embedded in a physically insecure technology is sufficient to secure the entire chip against algorithmic and physical attacks.**

***Keywords*—hardware redaction; logic locking; emerging technologies; hardware security; physical security; algorithmic security; satisfiability**

I. Introduction

As illustrated in Fig. 1, logic locking [1], [2] and hardware redaction [3] are logic design methodologies used to hide functionality and secure chips against malicious actors by disabling the chip if it is not programmed correctly after fabrication. While attackers may obtain the structural layout through an untrusted foundry or imaging, they cannot extract chip functionality without the correct key bitstream. Therefore, security of the chip hinges on the security of the key bitstream, which may be vulnerable to algorithmic [4] and physical attacks [5], [6].

While great strides have been taken toward securing the key against algorithmic attacks, it was recently demonstrated that any on-chip keys stored and/or used electrically are vulnerable to electrical imaging and other physical attacks [5], [6]. *If a chip is not secure against physical attacks, it is not secure at all*, regardless of its security against algorithmic attacks.

To provide security against physical attacks, physically secure circuits have been proposed with emerging device technologies that are not vulnerable to electrical imaging [7], [8]. The switching error rates of these emerging technologies, however, are too high for large-scale integration, limiting the size of circuits that can be secured against physical attacks.

Edwards et. al. conceptually proposed the use of multiple islands of physically secure emerging technology circuits within a large-scale CMOS system [8]. We demonstrate here that this hybrid approach for physical security also provides algorithmic security.

*Both authors contributed equally to this work.

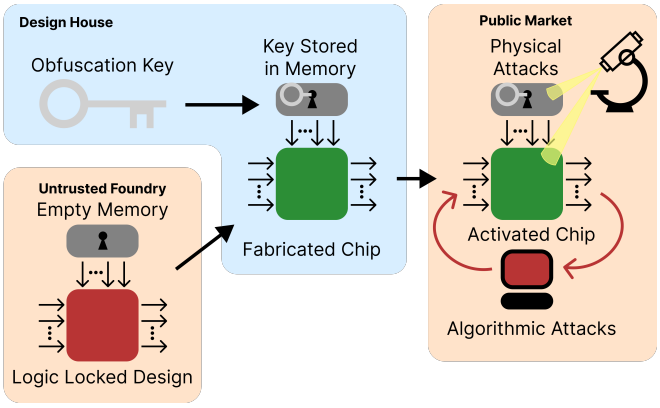

Fig. 1. Logic locking. Chip counterfeiting may be prevented by splitting the design into the transistor layout and an obfuscation key bitstream that unlocks the chip after fabrication. In order to replicate the design, the attacker must have both the layout and the obfuscation key. As it is generally assumed that the layout is not secure, security of the IP depends on the security of the obfuscation key stored in on-chip memory. Various imaging or algorithmic attacks may be used in attempts to obtain the key.

II. Background

A. Logic Locking and Hardware Redaction

Conventional CMOS logic locking involves the insertion of logic gates that cause the chip to function properly only if provided with correct obfuscation key inputs [9]. In emerging technologies, security can be achieved with polymorphic devices that can be programmed with a non-volatile key bit [10], [11], [12].

In hardware redaction, isolated portions of logic are substituted with embedded FPGAs, analogous to the logic-locked island approach of [8]. By obfuscating entire portions of the circuit rather than the individual gates conventionally used for logic locking, redaction increases algorithmic security and more effectively masks IP structure [3].

B. Attacks Against Redaction & Logic Locking

A malevolent attacker may attempt to overcome logic locking and hardware redaction through:

- Algorithmic attacks such as a satisfiability (SAT) attack, in which a design schematic and an activated chip (the oracle) are used to narrow down the key bitstream search space.
- Physical attacks that glean information about the key bitstream through physical signatures related to its storage or

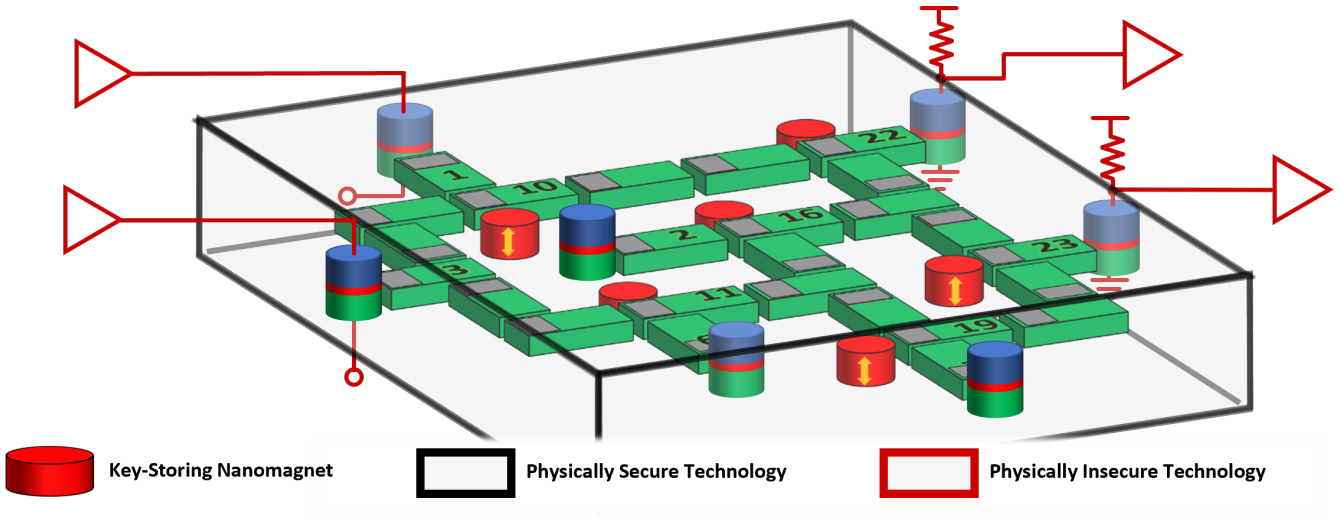

Fig. 2. Physically secure nanomagnet logic island embedded in physically insecure CMOS. As key-storing nanomagnets are embedded in the secure technology, attacks on the key must involve an algorithmic component.

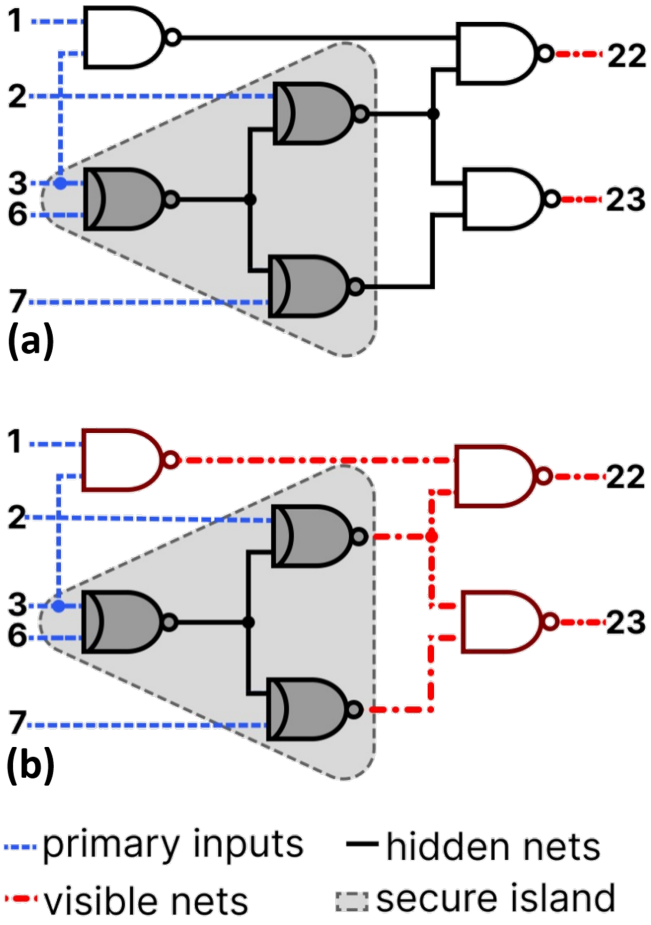

Fig. 3. (a) Classical SAT attack model. Primary inputs of the oracle are controllable and primary outputs are observable. (b) Physically enhanced SAT attack model. Primary inputs are controllable, and *all nets* not hidden within the physically secure islands are visible.

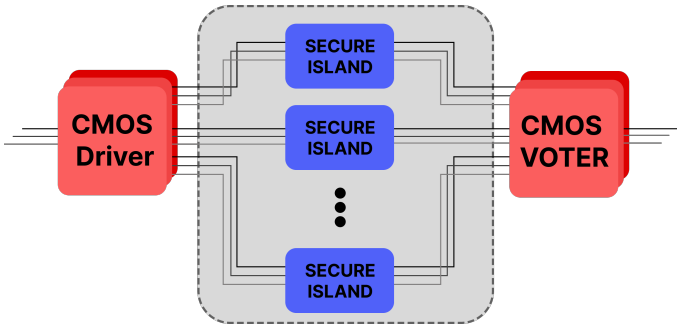

Fig. 4. Voting among replicated secure islands minimizes errors.

use, such as through electrical or magnetic imaging [5], [6], delayering [13], or power/delay analysis [14], [15].

C. Physically & Algorithmically Secure Hybrid Logic Systems

As illustrated in Fig. 2, Hassan et. al. used emerging technologies to demonstrate the first logic locking approach that is physically and algorithmically secure [7], but the circuit sizes are limited by high switching error rates [7], [8]. To enable large-scale circuits that are physically and algorithmically secure, "islands" based on these physically secure technologies can be used to logic lock or redact small portions of logic within a larger system that lacks physical security. With this island approach, the key is stored and used only within the physically secure islands, thus securing the key bitstream from physical attacks without suffering from large logical error rates.

D. Physically Enhanced Algorithmic Attack Model

A malevolent attacker may develop an attacking approach to discover the key through a combination of physical and algorithmic techniques: electrical imaging can be used to probe physically insecure nodes, thereby assisting the algorithmic attacks. Whereas the standard algorithmic attack model of Fig. 3(a) assumes that the oracle is a black box where only

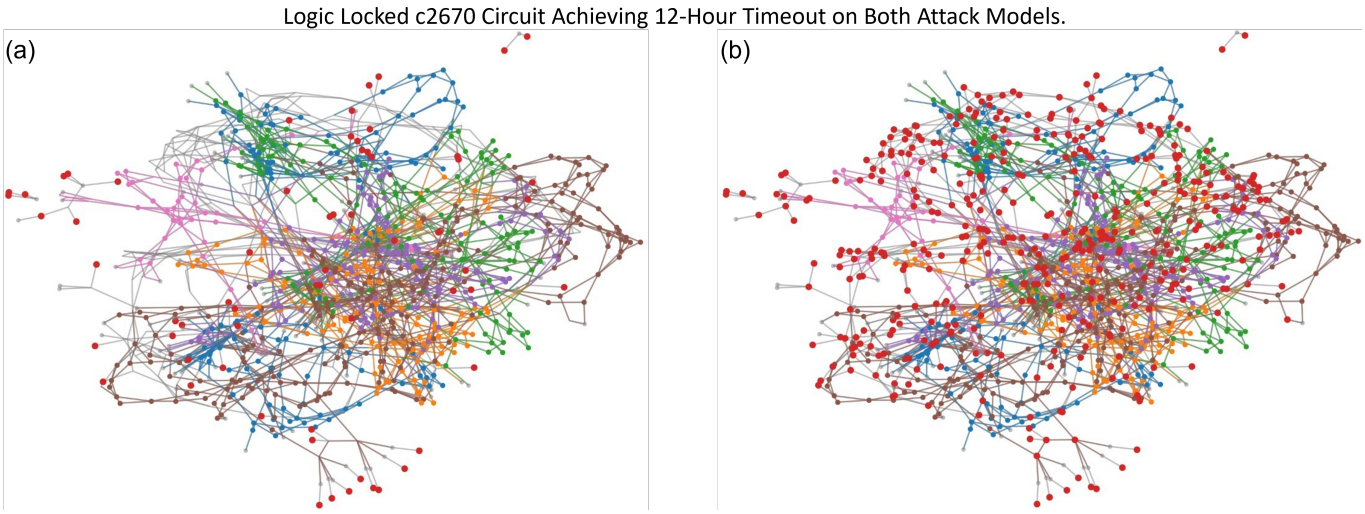

Fig. 5. Graph of logic-locked ISCAS '85 c2670 circuit with six islands of 150 gates each that successfully locked the circuit with a 12-hour timeout for both attack models. Edges represent input→output path of logic gates. Nodes correspond to wires. Colored nodes and edges correspond with physically secure islands (each island represented by a different color), whereas grey nodes and edges correspond with gates implemented in physically insecure CMOS. All NAND/NOR gates within an island are locked polymorphic gates storing one bit of the key bitstream. Large red nodes are visible to the attacker, with (a) illustrating a conventional SAT attack where only primary inputs and outputs are visible to the attacker while (b) illustrates a physically augmented SAT attack in which all wires not completely within a single island are visible.

primary outputs are visible, the enhanced attack model of Fig. 3(b) assumes that physically insecure internal nodes may be probed. Therefore, in the enhanced attack model, only nodes connecting gates which are entirely within secure islands are hidden from view of the attacker. This physically enhanced attack would require chip imaging after each solver iterationm, thereby significantly slowing the attack.

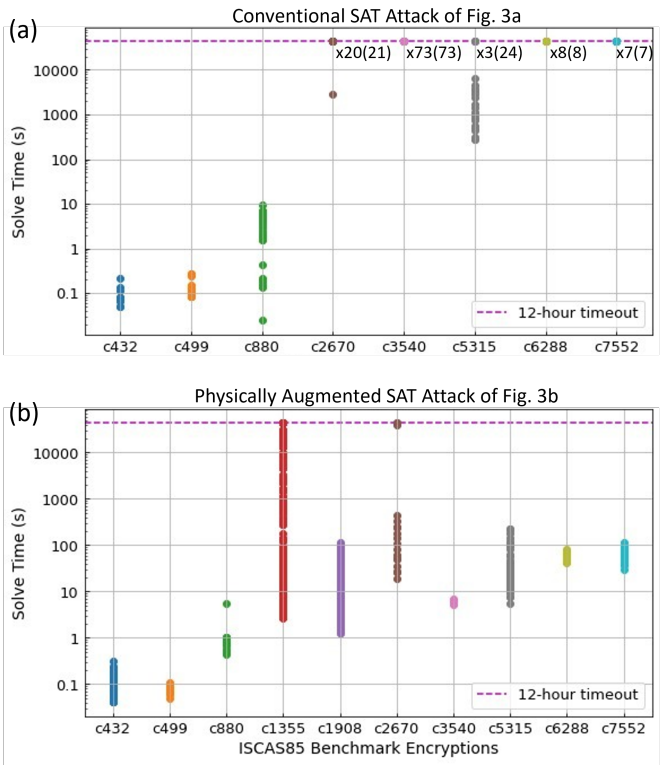

Fig. 6. SAT solve time of various logic-locked ISCAS '85 benchmark circuits with (a) the conventional SAT attack of Fig. 3a and (b) physically augmented SAT attack of Fig. 3b. Numbers correspond with the number of timeouts with the total number of attacks in parentheses.

III. Algorithmic Security with Physically Secure Islands

To demonstrate algorithmic security with physically secure islands, we performed algorithmic attacks under the generic SAT (Fig. 3a) and physically enhanced (Fig. 3b) attack models described above. Benchmark circuits from the ISCAS '85 suite [16] were logic locked with NAND/NOR polymorphism as in [7], with errors minimized through a voting scheme with duplicated islands. Assuming a gate error rate of 10^{-8} [17] and using three-to-one or five-to-one voting (illustrated in Fig. 4), we considered a reasonable island size between 100 and 1000 gates to match CMOS error rates of 10^{-12}. Gates inside secure islands were replaced with key-bit-programmable NAND/NOR polymorphic gates, and in the physically enhanced attack (Fig. 3b) all nets that are not entirely within discrete islands were treated as primary outputs.

A large number of logic-locked ISCAS '85 benchmark circuits [16] were attacked with the SAT solver under both attack models, and we were able to sufficiently secure some of the chips against both attack models so that they timed-out after 12 hours. Given the large design space, islands were selected according to the stochastic greedy algorithm described in the Methods. As illustrated in Fig. 6a, it is easier to secure the circuits against the conventional SAT attack model, as the solver has access to fewer observable nets. As depicted in Fig. 6b, we were able to achieve timeouts with some of the benchmark circuits on the physically enhanced attack model as well, which is remarkable considering islands were chosen randomly. While we believe that existing hardware redaction and logic locking techniques can be tailored to this augmented attack model to more predictably encrypt hybrid logic circuits, the success with randomized island selection

proves that small physically secure islands are indeed able to provide algorithmic security within a large hybrid system.

IV. Conclusions

Small islands that are physically secure are sufficient to provide algorithmic security within logic-obfuscated ICs, even against physically enhanced algorithmic attacks. Future efforts should tailor existing encryption algorithms for securing large hybrid circuits with physically secure islands.

Methods

Island Selection

The weighted search encryption algorithm begins by initializing two dictionaries outlined in Algorithm 1: "wire_map_inputs" and "wire_map_outputs". "wire_map_inputs" maps each wire to the set of gates that the wire drives, and "wire_map_outputs" maps each wire to the gate that drives it. Primary inputs are mapped to -1.

Algorithm 1: Map wires to corresponding gate outputs

```
for each gate do
    for each input wire connected to gate do
        connect input_wire and current_gate in
          wire_map_inputs dictionary
    end
end
for each gate do
    connect output_wire and current_gate in
      wire_map_outputs dictionary
end
for each primary input do
    connect output_wire and -1 wire_map_outputs
      dictionary
end
return wire_map_inputs, wire_map_outputs
```

The second portion of the algorithm seeds each island with a randomly chosen gate (see Algorithm 2). Each seed gate is added to the priority queue (pq) for its island, for processing in the third step.

Algorithm 2: Create island seeds by random choice

```
while total number of islands < desired total do
    pick a random gate
    if chosen gate has already been visited then
        continue
    end
    initialize new island with the chosen gate as the
      seed
end
return list of islands
```

The third portion of the algorithm (Algorithm 3), operates similar to Dijkstra's algorithm by growing each island by adding gates connected to the island until each island has reached the desired size. The algorithm prioritizes adding gates which will reduce the number of visible wires in the physically augmented SAT attack model of Fig. 3b. The algorithm achieves this by assigning each candidate gate a visibility score which is used for sorting the candidate gates in the priority queue. This visibility score is calculated according to Algorithm 4.

Algorithm 3: Minimum Visibility Search

```
create wire-gate maps (Algorithm 1)
create island seeds (Algorithm 2)
for the desired number of gates per island do
    for each island do
        if island priority queue is empty then
            continue to next island
        end
        set the current_gate to the candidate gate with
          minimum visibility score in priority queue
          and remove it from the queue
        visit current_gate and add it to the current
          island

        // add fan-in and fan-out wire's gates to priority
          queue
        for current_gate's input_wires do
            identify the fan_in_gate driving the
              input_wire using the wire_map_outputs
              dictionary
            if the wire is a primary input then
                continue to next wire
            end
            set visibility score for this gate using
              Algorithm 4
            add fan_in_gate and visibility score to
              island priority queue
        end
        for each fan_out_gate mapped to the current
          gate's output_wire do
            set visibility score for this gate using
              Algorithm 4
            add gate and visibility score to island
              priority queue
        end
    end
end
return islands
```

References

[1] J. A. Roy *et al.*, "Epic: Ending piracy of integrated circuits," in *DATE*, 2008.

[2] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *HOST*. IEEE, 2015.

[3] P. Mohan *et al.*, "Hardware redaction via designer-directed fine-grained efpga insertion," in *DATE*, 2021.

[4] J. Rajendran *et al.*, "Security analysis of logic obfuscation," in *DAC Design Automation Conference 2012*, 2012, pp. 83–89.

Algorithm 4: Calculate current_gate visibility

```
// visibility scores are calculated by subtracting new inputs added to island by gates and adding new outputs
for current_gate's wires do
    if current wire is a primary output then
        increment score by 1
        continue
    end
    if current wire is a primary input then
        decrement score by 1
        continue
    end

    // compare inside and outside visibility
    for each gate do
        if gate is current_gate then
            continue
        end
        if current_wire connected to gate then
            increment outside count by 1
        end
    end
    for each island do
        for gate in the island do
            if current_wire connected to gate then
                increment inside count by 1
            end
        end
    end

    // if inside and outside are different, there is visibility
    if outside count != inside count then
        if current_wire is an output wire then
            increment score by 1
        else
            decrement score by 1
        end
    else
        decrement score by 1
    end
end
return score
```


[5] S. Engels *et al.*, "The end of logic locking? a critical view on the security of logic locking." *IACR Cryptol. ePrint Arch.*, 2019.

[6] M. T. Rahman *et al.*, "The key is left under the mat: On the inappropriate security assumption of logic locking schemes." *IACR Cryptol. ePrint Arch.*, 2019.

[7] N. Hassan *et al.*, "Secure logic locking with strain-protected nanomagnet logic," in *Design Automation Conference*, 2021.

[8] A. J. Edwards *et al.*, "Physically and algorithmically secure logic locking with hybrid cmos/nanomagnet logic circuits," in *DATE*, 2022.

[9] S. Dupuis *et al.*, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*. IEEE, 2014, pp. 49–54.

[10] F. Parveen *et al.*, "Hybrid polymorphic logic gate with 5-terminal magnetic domain wall motion device," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017, pp. 152–157.

[11] S. Patnaik *et al.*, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 97–102.

[12] N. Rangarajan *et al.*, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *arXiv preprint arXiv:1811.06012*, 2018.

[13] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proceedings of the 48th Design Automation Conference*, 2011, pp. 333–338.

[14] M. Yasin *et al.*, "Security analysis of logic encryption against the most effective side-channel attack: Dpa," in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*. IEEE, 2015, pp. 97–102.

[15] ——, "Hardware security and trust: Logic locking as a design-for-trust solution," in *The IoT Physical Layer*. Springer, 2019, pp. 353–373.

[16] "ISCAS'85 Benchmark." [Online] Available: http://pld.ttu.ee/ maksim/benchmarks/iscas85/bench/.

[17] M. M. Al-Rashid *et al.*, "Effect of nanomagnet geometry on reliability, energy dissipation, and clock speed in strain-clocked dc-nml," *IEEE Transactions on Electron Devices*, vol. 62, no. 9, 2015.