Logic Synthesis and Verification Final Project Report - Synthesis for Security

Hsin-Tzu Chang
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei, Taiwan
r12943103@ntu.edu.tw

Wei-Shen Wang
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei, Taiwan
r12943102@ntu.edu.tw

***Abstract*—The burgeoning field of semiconductor manufacturing faces a significant threat from integrated circuit (IC) piracy, a pressing issue that demands immediate attention in IC design. This report introduces two classic defense strategies against reverse engineering: logic obfuscation and IC camouflaging. Our investigation involves a comprehensive survey of these techniques, including their foundational assumptions and algorithms. In addition to the theoretical exploration, we have implemented the algorithm for logic obfuscation and recorded the resultant design overheads.**

***Index Terms*—Hardware security, logic obfuscation, IC camouflaging, reverse engineering, piracy.**

I. Introduction

The global proliferation of Integrated Circuit (IC) systems has catalyzed remarkable innovations while simultaneously unveiling vulnerabilities, especially in the face of reverse engineering within the electronics industry. This intricate process involves disassembling the structure and function of the IC, risking unauthorized access to critical components. Reverse engineering paves the way for Intellectual Property (IP) theft, replication, or unauthorized use in competing products, underscoring the imperative for robust hardware security measures.

In response to these threats, the development of secure synthesis algorithms has become a critical focal point in safeguarding IC designs. These algorithms encompass various design levels, such as logic obfuscation at the gate level and IC camouflage at the design level. Sophisticated security measures aim to prevent reverse engineering attempts and strengthen the security of high-value IP. This approach significantly reduces the negative impact of piracy and unauthorized access, thereby preserving the integrity and confidentiality of critical IP in the IC design landscape.

II. Preliminaries

A. Logic obfuscation

Logic obfuscation keeps IC from being pirated by adding security on a gate-level aspect. According to [1], by inserting XORs and XNORs in the gate-level netlist, end-users are required to obtain the correct key bits to make the obfuscated netlist function correctly, as shown in Fig. 1 [2]. These inserted gates are referred to as *key-gate* in this report. Although [1] claims that all the key bits can only be brute-forced by attacker($O(2^{\text{number of key bits}})$ trials). Later in [2], the security level of inserting key-gate in netlist is being carefully analyzed. The study in [2] indicates that attackers are forced to perform brute-force attack only under certain key-gate structure. Hence, [2] proposes *smart logic obfuscation* to maximize the effort required by attackers. A detailed discussion of smart logic obfuscation is presented in Section III-A.

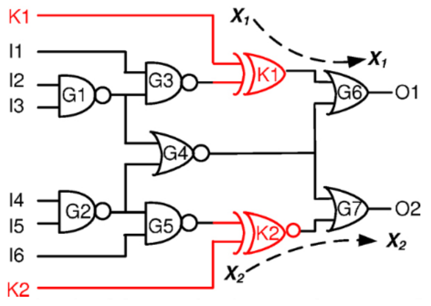


Fig. 1: Key-gate Insertion Example [2]

B. IC camouflaging

IC camouflaging strategically integrates camouflaged gates within a design to prevent netlist extraction at the layout level. Through the addition of dummy contacts, a set of standard cells can be manufactured with uniform layouts [4], hindering easy identification of the camouflaged gates' functionality by attackers. For instance, Fig. 2 [3] depicts layouts (a) and (b) representing distinct 2-input NAND and 2-input NOR cells, which are typically easily distinguishable from one another. However, employing camouflaged gate designs shown in (c) and (d) presents identical layouts, confounding potential attackers and hindering accurate gate-level netlist extraction.

Extracting a deceiving netlist forces attackers to interpret each camouflaged gate's functionality, resulting in an exponentially vast number of possible combinations. Misinterpreting any camouflaged gate may alter the circuit behavior significantly.

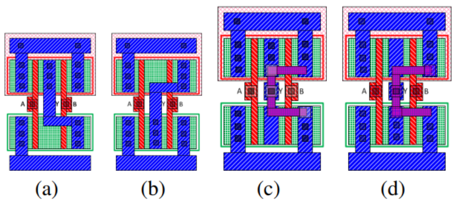


Fig. 2: Regular and camouflaged standard cell layouts of 2-input NAND and 2-input NOR [3].

In the context of [3], the threat model is clearly defined. Firstly, the attackers can delayer an IC and extract the gate-level netlist using advanced image processing techniques. Secondly, they can differentiate between standard cells and camouflaged ones. Lastly, the attackers possess knowledge about potential functions represented by camouflaged gates. Consequently, IC camouflaging focuses on strategically choosing gates for camouflage within the design, significantly increasing reverse engineering complexity. Detailed explanation of the gate selection algorithm will be in Section III-B.

III. Algorithm and Framework

A. Logic obfuscation

1) Assumption of the attacker

It is assumed that the attackers can determine the locations of the key-gates inserted in the netlist. Furthermore, the attackers are presumed to have the capability to purchase a fully functional IC from the market [1] and to brute-force all *PI(Primary Input)s-to-PO(Primary Output)s* relationships. However, they are not able to observe any internal value of netlist other than PIs and POs.

2) Key-Gate Structures

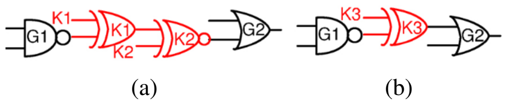
This section presents an analysis of different key-gate structures. Table I provides a brief overview of the relationship between various key-gate structures and their corresponding security level.

Security Level	Key-Gate Structures
Ineffective	Runs of key-gates
Isolated/Unsafe	Isolated key-gate
Mutable/Medium	Dominating key-gate Concurrent key-gate Sequential key-gate
Non-Mutable/Safe	Non-Mutable key-gate

TABLE I: Security level of Key-Gate Structures

a) Runs of key-gates

If an output of key-gate is directly connected to the input of another key-gate as shown in Fig. 3a, the structure is identified as run of key-gates. [1] The run of key-gates can contain more than 2 key-gates. However, Table I categorizes this structure as ineffective, as attackers can simplify it to an equivalent netlist with a single key-gate, depicted in Fig. 3b. Regardless of the number of gates in a run of key-gates, this structure neither increases the complexity for an attacker to reverse engineer the netlist nor does it improve the circuit's efficiency, instead leading to increased delay, area, and power consumption.



(a) (b)

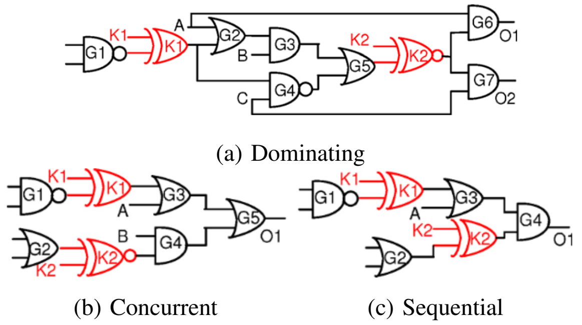
Fig. 3: Runs of key-gates [2]

b) Isolated key-gate

An isolated key-gate is characterized by the absence of any path connecting it to other key-gates. [2] It can be solved without considering the existence of any other key-gate. Fig. 1 is an example in which both K1 and K2 are isolated. In solving K1, attackers can ignore components G2, G3, G7, O2, and K2, focusing solely on the relationship between I1, I2, I3, and O1. The method for resolving K2 is analogous to that of K1. This type of key-gate structure is deemed unsafe as it fails to effectively increase the total key-length of the obfuscated circuit.

c) Dominating key-gate

Dominating key-gate falls within the category of mutable key-gate. Fig. 4a illustrates an example where K2 dominates K1(K2 is on every path for K1 to reach any POs). Hence, if A=1, the effect of K1 will be *muted.* In this context, *muted* refers to a key-bit value that is overshadowed or controlled by another value [2]. Consequently, when K1 is muted, K2 can be resolved without considering K1's effect on the POs.



(a) Dominating

(b) Concurrent (c) Sequential

Fig. 4: Mutable Key-gate [2]

d) Concurrent key-gate

Concurrent key-gate involves a specific structure where two key-gates can be muted respectively when the attackers are trying to solve another key-bit. This structure is exemplified in Fig. 4b, where the two key-gates converge at the same PO. In Fig. 4b, when A is set to 1, K1 becomes muted, whereas if B is set to 0, K2 is muted.

e) Sequential key-gate

While the sequential key-gate is deemed marginally safer compared to the concurrent key-gate structure, it still belongs to the category of mutable key-gate. As illustrated in Fig. 4c, when A is set to 1, K1 becomes muted. This structure is considered slightly safer than concurrent key-gates because unlike the latter, K2 cannot be muted in this situation. Consequently, the attackers are compelled to mute K1 and subsequently resolve K2, which effectively narrows down their options for attacking the netlist.

f) Non-Mutable key-gate

The non-mutable key-gate represents the ideal structure for the defenders, as it compels the attackers to resort to brute-force methods to resolve the key-bit. In this structure, the effective length of the key remains intact and cannot be reduced through any known method. Fig. 5 demonstrates this concept, where for K1 to be muted, the other input of G4 needs to be controlled to 1. However, this input is governed by K2. Attempting to mute K2 leads to a similar impasse, where controlling the corresponding key is not feasible. As a result, the attackers are left with no alternative but to exhaustively guess the key-bits.

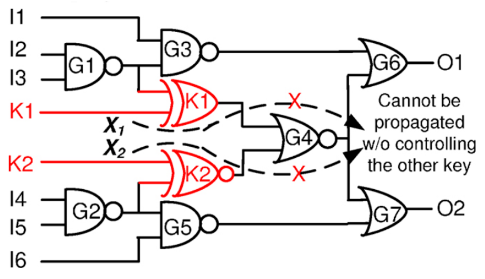


Fig. 5: Non-mutable key-gate [2]

3) Smart key-gate Insertion

We organize the defense algorithm in [2] into a flowchart(Fig. 6). Initially, the algorithm randomly inserts 10% of the *number of total key-gates* into the gate-level netlist. In this context, *number of total key-gates* refers to approximately 5% of the netlist's total gate count, a proportion that can be adjusted based on user specifications. Following the random insertion, the algorithm updates the key-gate locations and evaluates whether the number of key-gates is sufficient. If the current number meets the requirement, the algorithm concludes. Otherwise, the algorithm proceeds to calculate the potential *Hardness* for each non-key-gate. Here, G represents a non-key-gate. The metric assigns 0 point for isolated key-gates, 1 point for mutable key-gates, and 2 points for non-mutable key-gates, with the possibility of adjustment based on user requirements. After computing the *Hardness*, the gate with the highest *Hardness* is selected and integrated into the netlist as a new key-gate, followed by an update to the current key-gate locations. This algorithm ensures that runs of key-gates are avoided by not considering locations directly connected to any existing key-gate as potential sites for new key-gates.

$$Hardness(G) = \sum_{Ki} metric(G, Ki).$$

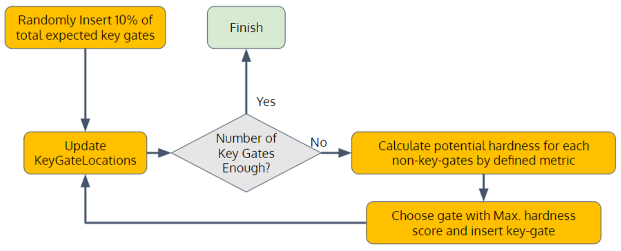

Randomly Insert 10% of total expected key gates
Finish
Yes
Update KeyGateLocations
Number of Key Gates Enough?
No
Calculate potential hardness for each non-key-gates by defined metric
Choose gate with Max. hardness score and insert key-gate

Fig. 6: Logic Obfuscation flow chart [2]

B. IC camouflaging

This section commences with an overview of the reverse engineering strategy employed in IC design. Subsequently, we introduce the IC camouflaging algorithm that counteracts this strategy. The primary aim of this algorithm is to substantially increase the complexity of the reverse engineering process. To provide a clear understanding, the potential function list for a 2-input camouflaged gate within the scope of our discussion is explicitly defined. The functions considered are {XOR, NAND, NOR}.

1) Attacker's Strategy

After extracting the deceiving netlist, the attackers must discern the functionality behind each camouflaged gate. Table II outlines a verification method: assigning the input pattern '00' identifies an XOR gate, while '01' or '10' distinguishes a NOR gate. Notably, input pattern '11' consistently results in an output of 0 across all three functions. Therefore, for a single camouflaged gate, if input patterns can be justified to '00' and '01' (or '10'), and the output can propagate to POs, the functionality of the camouflaged gate can be identified.

Input	XOR	NAND	NOR
00	0	1	1
01	1	1	0
10	1	1	0
11	0	0	0

TABLE II: Truth Table for Potential Function List

Camouflaged gates are termed as *isolated camouflaged gates* if there are no interference between each of the two camouflaged gates and their functionalities can be identified independently. In Fig. 7, both C1 and C2 exemplify such gates. Assigning the input pattern '01101110' justifies inputs as '00' for both gates, revealing XOR functionalities at POs O1 and O2. Similarly, using '11101111' justifies inputs as '01', confirming NOR functionalities at POs O1 and O2. This independent identification establishes C1 and C2 as isolated camouflaged gates.

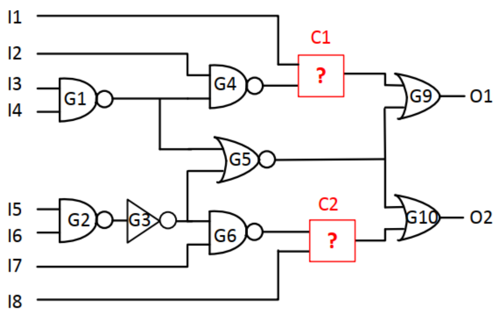
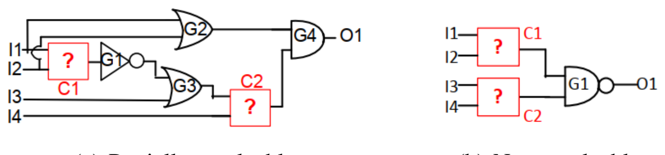


Fig. 7: Isolated camouflaged gates [3].

In specific scenarios, interference among camouflaged gates leads to partial resolution of their functionalities. In Fig. 8a, input pattern 'x110' justifies C2's input as '10', sensitizing its output to propagate to O1. If O1 is 0, C2 is identified as NOR. Otherwise, determining C2 requires knowledge of C1's functionality. Likewise, resolving C1 depends on C2's functionality, making C1 and C2 *partially-resolvable gates*.

In Fig. 8b, strong interference between C1 and C2 prevents their independent resolution. To resolve C1, C2's output must be a non-controlling value(1), which is impossible without knowing C1's functionality. The same applies to resolving C2. Hence, C1 and C2 are labeled as *non-resolvable gates*. Resolving these gates necessitates a brute force approach—testing various functionalities and validating correctness through extensive logic simulation.



(a) Partially-resolvable (b) Non-resolvable

Fig. 8: Different security level of camouflaged gates [3]

Consequently, the attackers' strategy starts with identifying and resolving isolated camouflaged gates, requiring computational effort of $O(m)$, where m represents the count of isolated camouflaged gates. Next, the attackers try to resolve partially-resolvable gates. Finally, a brute force approach tackles non-resolvable gates. For a set of n pairwise non-resolvable gates, the potential functionality combinations reach 3^n, demanding extensive computational resources and effort in reverse engineering due to the vast search space.

2) Smart Camouflaging Algorithm

The defenders strategically choose specific circuit gates to be replaced with camouflaged ones. While camouflaging all gates might seem ideal for security, the design overhead makes it inefficient. Hence, a more practical approach is to camouflage certain amount of gates.

Two metrics gauge IC camouflaging effectiveness. Firstly, it aims to maximize complexity, making accurate interpretation of camouflaged gates challenging for attackers. Secondly, any misinterpretation should significantly impact the circuit's behavior.

In line with the attackers' strategy mentioned earlier, the primary objective in gate selection is to maximize the largest group size of non-resolvable camouflaged gates, aligning with the first metric. As the sole objective, an iterative strategy involves consistently choosing gates from the largest group of pairwise non-resolvable camouflaged gates.

For the second objective, an evaluating metric *Output Corruptibility (OC)* of a gate G is introduced:

$$OC(G) = \sum_{\text{all input patterns}} \{\text{\#incorrect PO if } G \text{ is incorrectly interpreted}\}.$$

Similarly, an intuitive selection algorithm is to choose the gates with highest OC.

However, experimental findings in [3] suggest that algorithms focusing on a single objective result in poor security performance. For instance, the non-resolvable-gate-based algorithm demands exceedingly high computational effort to resolve all camouflaged gates, despite many camouflaged gate outputs rarely impacting the POs, exerting minimal influence on circuit behavior. Therefore, an integrated *Smart Camouflaging Algorithm* is introduced.

This algorithm begins by computing Output Corruptibility (OC) for each gate and selecting the one with the highest OC. Subsequent iterations identify non-resolvable gates based on previously selected camouflaged gates, opting for the gate with the highest OC among them. Empirical evidence in [3] supports the superiority of this algorithm across both evaluation metrics.

The effort required to resolve camouflaged gates remains similar to the non-resolvable-gate-based algorithm since the selected gates for both are non-resolvable. Simultaneously, the algorithm's impact from misinterpretation aligns with the output-corruptibility-based approach by selecting gates with the highest OC in each iteration.

In conclusion, the integrated *Smart Camouflaging Algorithm* achieves an exceptional balance between the two objectives, resulting in a layout of remarkable security and performance.

IV. Experimental Results

A. Our Implementation

We re-implement logic obfuscation(key-gate insertion) in this final project. We adopt a simplified metric for key-gate evaluation: isolated key-gates score 0, and non-isolated key-gates score 1, treating mutable key-gate and non-mutable key-gate the same. The methodology for identifying an isolated key-gate is not explicitly detailed in [2], so we develop our own determination method. For two gates K1 and K2 to be considered non-isolated, they must have identical reachable POs. Conversely, if a PO, say O1, is only reachable by one key-gate, say K1, then K1 is deemed isolated. This is because K1 can be resolved independently of K2 when analyzing the relationships between PIs and O1.

B. Experimental Setup

In our study, we employed five combinational circuits from the ISCAS'85 benchmark suite as our primary benchmarks [5] [6]. Both the original and the obfuscated netlists are processed using the ABC tool. They are then converted into And-Inverter Graphs (AIG) using the *strash* command within ABC. In this context, the area is quantified by the number of AND gates, whereas the delay is represented by the depth of these gates. We observed that key-gate insertion result in around 20% of trade-off in both area and delay.

	Original		Obfuscated		Overhead	
Circuit	Area	Delay	Area	Delay	Area	Delay
c880	325	25	397	31	18.14%	19.35%
c1355	494	25	584	33	15.41%	24.24%
c3540	1034	41	1097	50	5.74%	18.00%
c5315	1774	37	2146	38	17.33%	2.63%
c6288	2337	120	2445	122	4.42%	1.64%

TABLE III: Area and Delay calculated by ABC

V. Conclusion

This final project comprehensively surveys and introduces two significant papers in the field of Synthesis for Security: **Security Analysis of Logic Obfuscation** [2] and **Security Analysis of Integrated Circuit Camouflaging** [3]. This final project primarily focuses on re-implementing the algorithm presented in [2]. Additionally, we evaluate the impact of key-gate insertion on the area and delay of circuits, highlighting the trade-offs between security enhancement and circuit cost. Looking ahead, integrating IC camouflage with key-gate insertion could offer a higher level of security, presenting the attackers with more complex challenges. Moreover, future work could also strategically avoid the insertion of key-gates on critical path to avoid delay-related costs. To conclude, this project offers insights into Security for Synthesis and guides beginners to the field.

VI. Job Division

Wei-Shen Wang:

- Paper survey and re-implementation on the work **Security analysis of logic obfuscation** [2].
- Transform gate-level netlist into Berkeley-ABC readable format for area and delay evaluation.

Hsin-Tzu Chang:

- Paper survey on the work **EPIC:ending piracy of integrated circuits** [1] and **Security analysis of integrated circuit camouflaging** [3].
- Parsing of the ISCAS'85 benchmark suite [5] [6].

References

[1] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. 2008. EPIC: ending piracy of integrated circuits. In Proceedings of the conference on Design, automation and test in Europe (DATE). Association for Computing Machinery, New York, NY, USA, 1069–1074.

[2] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri. 2012. Security analysis of logic obfuscation. Design Automation Conference (DAC), San Francisco, CA, USA, pp. 83-89

[3] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security analysis of integrated circuit camouflaging. In Proceedings of the ACM SIGSAC conference on Computer & communications security (CCS). Association for Computing Machinery, New York, NY, USA, 709–720.

[4] L. W. Chow, J. P. Baukus, and W. M. Clark. Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide. U.S. Patent 7294935, Jul. 25, 2002.

[5] Brglez, Franc. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. Proc. Intl. Symp. Circuits and Systems.

[6] https://github.com/jpsety/verilog_benchmark_circuits

[7] ABC: System for Sequential Logic Synthesis and Formal Verification