Layout Recognition Attacks on Split Manufacturing

Wenbin Xu, Lang Feng, Jeyavijayan (JV) Rajendran, and Jiang Hu
Department of Electrical and Computer Engineering
Texas A&M University, College Station, Texas
wbxu@tamu.edu,flwave@tamu.edu,jeyavijayan@tamu.edu,jianghu@tamu.edu

ABSTRACT

One technique to prevent attacks from an untrusted foundry is split manufacturing, where only a part of the layout is sent to the untrusted high-end foundry, and the rest is manufactured at a trusted low-end foundry. The untrusted foundry has front-end-of-line (FEOL) layout and the original circuit netlist and attempts to identify critical components on the layout for Trojan insertion. Although defense methods for this scenario have been developed, the corresponding attack technique is not well explored. For instance, Boolean satisfiability (SAT) based bijective mapping attack is mentioned without detailed research. Hence, the defense methods are mostly evaluated with the k-security metric without actual attacks. We provide the first systematic study, to the best of our knowledge, on attack techniques in this scenario. Besides of implementing SAT-based bijective mapping attack, we develop a new attack technique based on structural pattern matching. Experimental comparison with bijective mapping attack shows that the new attack technique achieves about the same success rate with much faster speed for cases without the k-security defense, and has a much better success rate at the same runtime for cases with k-security defense. The results offer an alternative and practical interpretation for k-security in split manufacturing.

CCS CONCEPTS

• **Security and privacy → Hardware reverse engineering**; *Hardware attacks and countermeasures*;

KEYWORDS

Hardware security, split manufacturing, layout recognition attack

ACM Reference Format:
Wenbin Xu, Lang Feng, Jeyavijayan (JV) Rajendran, and Jiang Hu. 2019. Layout Recognition Attacks on Split Manufacturing. In *24th Asia and South Pacific Design Automation Conference (ASPDAC '19), January 21–24, 2019, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3287624.3287698

1 INTRODUCTION

Split manufacturing is a security technique against untrusted foundries. By having only front-end-of-line (FEOL) layers manufactured at an untrusted foundry while the back-end-of-line (BEOL) is fabricated at a trusted foundry, attackers in the untrusted foundry do not have complete information to perform attacks such as Trojan insertion, piracy, and overproduction [2, 14, 21]. The same principle can be applied to 3D ICs, where different dies are manufactured by different foundries, since each 3D IC contains two or more independently manufactured ICs which are vertically stacked on top of each other [3]. Despite the security enhancement, split manufacturing still has a significant risk of being successfully attacked [8, 12, 19].

There are two attack scenarios in split manufacturing. (i) The attacker at FEOL foundry does not have circuit netlist and attempts to reverse engineering the entire design for stealing intellectual property, and conducting piracy and overproduction; (ii) The attacker has circuit netlist and tries to recognize critical components on the layout for inserting Trojans. Most existing attack models are proposed for (i), such as the work in [19], and there are also defense methods against the attack models, such as placement perturbation [19] and routing perturbation [20].

With access to the netlist, the attack in scenario (ii) may seem to be easier than that in scenario (i). However, (ii) requires much more understanding of the layout design than (i). More specifically, the attack in (i) only needs to reproduce the overall design without understanding its layout details while (ii) must recognize layout components and match them exactly to the netlist. With BEOL information missing, the layout-to-netlist matching is not trivial at all. However, the investigation on scenario (ii) is mostly restricted to only defense techniques. For example, a previous work [3] proposed a method using a metric called k-security to protect the layout by obfuscation. The metric k-security means that for every group of components of which the connections are visible in FEOL, it will have at least another k-1 groups which are identical to it logically. They also proposed a SAT-based algorithm [3] for selecting wires

Table 1: Comparison of different attack methods.

	SAT bijective [3]	Network-flow attack [19]	Structural pattern matching
Target to attack	Cells	Wires	Cells
Use of structural information	√	×	√
Use of design convention	×	√	√
Knowledge of circuit netlist	√	×	√
Recovery of BEOL connections	×	×	√

Table 2: Comparison of different defense methods.

	Placement perturbation [19]	Routing perturbation [20]	K-security [3]
Modification on placement	√	×	√
Modification on routing	×	√	√
Defense for physical attack	√	√	√
Defense for logical attack	×	×	√

ASPDAC '19, January 21–24, 2019, Tokyo, Japan
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3287698

to be manufactured in BEOL to achieve the k-security. Moreover, another work [7] improves the algorithm in [3] and realize a shorter runtime for achieving k-security. More details about the comparison of different typical attack methods and that of different typical defense methods are shown in Table 1 and 2.

To the best of our knowledge, this work provides the first systematic study on the attack in split manufacturing when untrusted foundry has access to netlist and intends to insert Trojans. Our work focuses on recognizing critical components in FEOL layouts rather than actually inserting Trojans. We propose a new attack technique using structural pattern matching assisted by hints from design conventions. We also implemented the SAT-based bijective mapping attack [3]. The experiment results show that the structural pattern matching attack is more efficient than the bijective mapping attack. For split manufacturing without k-security defense, the proposed attack techniques usually achieve near 100% success rate. They are also applied to designs with k-security defense to confirm the effectiveness of such defense. Moreover, the evaluation by actual attacks reveals richer and more tangible interpretation for k-security in split manufacturing. The main contributions of this paper are:

- We proposed an attack method based on structural pattern matching for layout recognition. Our attack method achieves shorter runtime and/or better performance than the SAT-based bijective mapping attack for circuits with and without k-security defense.
- The attack method we proposed uses both logical information and hints from design conventions, which are seldom considered at the same time within a same attack before.
- Our work on attack techniques provides an alternative way to evaluate the defense of split-manufactured ICs for Trojan insertion.

2 PRELIMINARY

2.1 Attack Scenario

In this work, we consider the scenario that an attacker is at an untrusted FEOL foundry and intends to insert Trojan into split-manufactured ICs. The attacker needs to understand circuit functions in the FEOL layout so that effective Trojan site can be decided. The attacker is assumed to know the following information:

1. Layout of transistor and FEOL metal layers.
2. Entire circuit netlist or part of the netlist that is related to Trojan insertion. The attacker may obtain this information by stealing from design companies or collaborating with an observer in design stage [3, 7].
3. The technology library containing information about logic gates: layout structure, delay, capacitance load, wire capacitance and design specifications.

2.2 Related Work

Split manufacturing was proposed to improve IC security against reverse engineering and Trojan insertion [4, 6]. There are several research works showing the benefits of split manufacturing on digital ICs [9, 10, 15–17] and analog ICs [1]. The concept of split

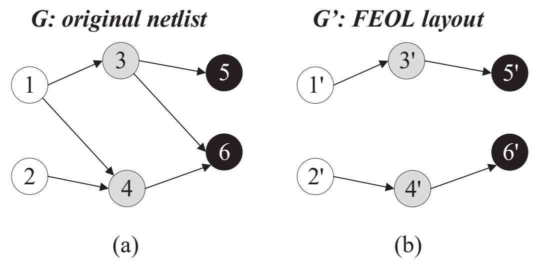


Figure 1: Graph representations of (a) circuit netlist and (b) its FEOL layout. Each vertex indicates a logic gate, whose logic type is represented by grayscale.

manufacturing can be extended to 3D or 2.5D IC designs such that different dies are manufactured at different foundries [18].

For the attack scenario where attackers only have incomplete designs, several works provide different techniques to restore missing BEOL wires so that the design can be reverse engineered [8, 12, 19]. Their attack is evaluated with a number of correctly restored wires even though the functions of related layout components are still unknown. To defend against such kind of attack, several strategies based on placement or routing are introduced [11, 19, 20]. Different from heuristic approaches for enhancing security, the work in [13] proposed a theoretical model based on information theory.

The work in [3] shows a different attack scenario where attackers have access to circuit netlist and intend to recognize layout for Trojan insertions. It mentions SAT-based bijective mapping attack without detailed elaboration. It proposes the concept of k-security, which leads to k identical options when attackers attempt to recognize layout. Indeed, this approach causes up to 200% area overhead. The recent work [7] proposes to restrict k-security to a small and critical portion of the circuit so that the overhead can be reduced. It inserts dummy cells to obfuscate the design further and enhance the security. However, the defense techniques in both [3] and [7] are evaluated by k-security without actual attacks.

Although the overall attack and defense in split manufacturing have been recently studied, there is no investigation on layout recognition attack driven by Trojan insertion, to the best of our knowledge.

2.3 SAT-based bijective mapping

SAT-based bijective mapping was briefly mentioned as a Trojan-driven attack to split manufacturing [3]. Generally, either circuit netlist or FEOL layout can be modeled as a graph, where each vertex represents a logic gate, and edges indicate wire nets. Normally, the netlist graph and its corresponding FEOL layout graph should have the same vertices, while the netlist graph contains more BEOL edges that are not available in the layout graph. Given a netlist graph $G = (V, E)$ and its layout graph $G' = (V', E')$ as shown in Figure 1, the goal of layout recognition attack is to find the vertex $v \in V$ that corresponds to each vertex $v' \in V'$. In bijective mapping, such correspondence is designated by Boolean variables

$$\phi_{ij} = \begin{cases} 1, & \text{if } v_i \in V \text{ can be mapped to } v'_j \in V' \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Then, the bijective mapping should be subject to three constraints: 1) mapping v in G to v' in G', 2) mapping v' in G' to v in G, and 3) mapping e' in G' to e in G.

Thus, the bijective mapping problem is reduced to a SAT instance [3]. By solving it via SAT solver, one can obtain one possible assignment for the set of ϕ_{ij}. When multiple assignments can be found, SAT or bijective mapping alone cannot decide which is the correct one. As such, an arbitrary feasible assignment solution is taken.

In the example shown in Figure 1, we can construct the SAT constraints as

$$\begin{aligned} F_1 =&(\phi_{11}\overline{\phi}_{12} + \overline{\phi}_{11}\phi_{12})(\phi_{21}\overline{\phi}_{22} + \overline{\phi}_{21}\phi_{22})(\phi_{33}\overline{\phi}_{34} + \overline{\phi}_{33}\phi_{34}) \\ &(\phi_{43}\overline{\phi}_{44} + \overline{\phi}_{43}\phi_{44})(\phi_{55}\overline{\phi}_{56} + \overline{\phi}_{55}\phi_{56})(\phi_{65}\overline{\phi}_{66} + \overline{\phi}_{65}\phi_{66}) \end{aligned} \quad (2)$$

$$\begin{aligned} F_2 =&(\phi_{11}\overline{\phi}_{21} + \overline{\phi}_{11}\phi_{21})(\phi_{12}\overline{\phi}_{22} + \overline{\phi}_{12}\phi_{22})(\phi_{33}\overline{\phi}_{43} + \overline{\phi}_{33}\phi_{43}) \\ &(\phi_{44}\overline{\phi}_{34} + \overline{\phi}_{44}\phi_{34})(\phi_{55}\overline{\phi}_{65} + \overline{\phi}_{55}\phi_{65})(\phi_{56}\overline{\phi}_{66} + \overline{\phi}_{56}\phi_{66}) \end{aligned} \quad (3)$$

$$\begin{aligned} F_3 =&(\phi_{11}\phi_{33} + \phi_{21}\phi_{43})(\phi_{12}\phi_{34} + \phi_{22}\phi_{44}) \\ &(\phi_{33}\phi_{55} + \phi_{43}\phi_{65})(\phi_{34}\phi_{56} + \phi_{44}\phi_{66}) \end{aligned} \quad (4)$$

$$F = F_1 \wedge F_2 \wedge F_3. \quad (5)$$

By solving Equation (5) via SAT solver, one satisfiable assignment is $q_1 = \phi_{11}\overline{\phi}_{12}\overline{\phi}_{21}\phi_{22}\phi_{33}\overline{\phi}_{34}\overline{\phi}_{43}\phi_{44}\phi_{55}\overline{\phi}_{56}\overline{\phi}_{65}\phi_{66}$.

The SAT-based bijective mapping is straightforward, but it still has several drawbacks:

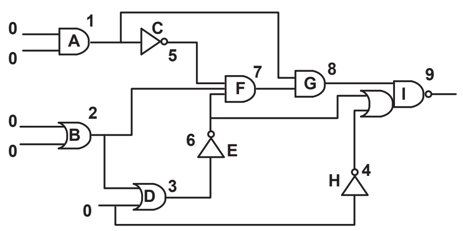
(1) **Recognition capability**: When multiple feasible mapping solutions exist, it does not provide any clue on which one is the truly correct layout recognition. Hence, such attack can easily fail on circuits with k-security protection.
(2) **Scalability**: SAT is a well-known NP-complete problem. Therefore, a solver finds feasible solution using exponential time in the worst case. Despite the tremendous progress on SAT solving techniques, the fundamental scalability challenge is not solved.
(3) **Flexibility**: Even for small circuits, the SAT-based bijective mapping can be effective only when it applies with complete netlist. Even if only a small portion of the layout needs to be recognized, it is applied to the entire circuit.

3 LAYOUT RECOGNITION ATTACK BY STRUCTURAL PATTERN MATCHING

We propose a new attack technique to recognize split manufactured layout in a more scalable manner than the SAT-based bijective mapping. Our technique is inspired by structural pattern matching [22], a technology mapping technique that determines which library cell can implement a subfunction in a given Boolean network. This is similar to the attack scenario described in Section 2.1, where one needs to match components in FEOL layout with those in the netlist. However, there are several significant differences between the pattern matching in the attack and that in technology mapping.

- In attacking split manufacturing, the netlist is to match with *incomplete* layout where BEOL information is not available. This is quite different from technology mapping and much more difficult to handle.
- Technology mapping is to match library cells, which are typically small. However, security attack is to match larger subcircuit or entire circuit. Such difference entails different data representation approaches.
- Technology mapping intends to map one library cell with many parts in a circuit design. However, the security attack is to identify a unique matching between netlist and layout. Therefore, a partial matching in a local region is often inconclusive.

Overall, the pattern matching in the layout recognition attack is much more challenging than that in technology mapping. We develop techniques to improve the attack by exploiting hints from design conventions.



(a)

Input 0	Index
0	4
1	5
3	6

INV

Input 0	Input 1	Index
0	0	1
1	7	8

AND2

Input 0	Input 1	Index
0	0	2
0	2	3

OR2

Input 0	Input 1	Input 2	Index
2	5	6	7

AND3

Input 0	Input 1a	Input 1b	Index
8	4	6	9

OAI21

(b)

Figure 2: Example of the pattern table: (a) circuit netlist and (b) its pattern tables associated with input pins of logic gates and indices.

3.1 Pattern Table

The pattern table is to represent logic and structural relationship in a Boolean network, and plays a similar role as the pattern table in pattern matching for technology mapping [22]. The format of our pattern table is the sparse matrix of the pattern table used in [22]. A pattern table is generated from a circuit netlist according to different gates, for example, $AND2$ gate and $OR2$ gate have separated pattern tables. The elements in the pattern table are pattern indices. Each logic gate is assigned with a pattern index. The index of a gate g is uniquely associated with its gate type and indices of its fanin gates. Figure 2 shows an example of a circuit netlist and its pattern tables. In this example, gate G is represented by pattern $8(AND2, 1, 7)$ where 8 is its pattern index, and 1 and 7 are indices of its fanin gates. Such information will be stored as the second row of $AND2$ pattern table in Figure 2(b). For an $AND2$ gate, the logic function of its two inputs are identical. This is represented by removing the solid lines between the columns of input 0 and input 1. Otherwise, if two inputs are not identical, like input 0 and input 1a of $OAI21$ gate in Figure 2, we used the solid line to separate the columns of input 0 and input 1a. If there is no solid line between two columns, that means the order of input pattern indices does not matter. Given a complete or partial circuit netlist, we generate pattern tables for all the gates through a topological order traversal.

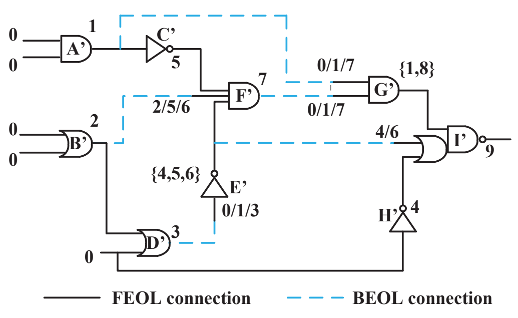


Figure 3: Matching cells with FEOL layout by using pattern tables (solid line: FEOL connection, dashed line: BEOL connection).

There is a difference in handling pattern indices between matching an entire circuit and a subcircuit. When matching an entire circuit, we assume that circuit I/O pins have already been matched to put more focus on matching of gates. In this case, each of corresponding I/O vertices in the netlist has its own unique index. When to match only a subcircuit, we cannot assume the knowledge of those I/O nodes. As such, their indices are uniformly 0 like in technology mapping [22], which are treated as don't care. By doing so, we only match the internal logic structure and function of the subcircuit.

3.2 Matching a Subcircuit with FEOL Layout

In this section, we describe how to match a subcircuit with FEOL layout. For an entire circuit represented by a graph G, a subcircuit can be treated as a subgraph $G' \subset G$. By the pattern table generation defined by Section 3.1, each $v' \in G'$ should be annotated with a pattern index.

For FEOL layout, a pre-processing is performed to identify logic gates from transistors according to the cell library. Next, we attempt to find pattern index for each gate in the layout. The pattern indices of all layout gates are initialized to 0. Later, if a layout gate is successfully matched with a node in the netlist, its index is set to be the same as that in the netlist. Figure 3 shows the layout for the circuit netlist of Figure 2, in which FEOL layers contain all the gates and connections in solid lines while BEOL layers include the connections in dashed lines. In this example, the pattern table of A' is $(AND2, 0, 0)$, which matches with the $1(AND2, 0, 0)$ in the netlist. Therefore, layout gate A' is also annotated with $1(AND2, 0, 0)$.

The key challenge here is that the layout information is not complete. In Figure 3, the dashed lines indicate BEOL connections, which are invisible to the attacker. For example, one input connection of gate F' is at BEOL. In this case, our attack keeps a set of candidate possibilities and prune them according to the pattern tables and simple logic reasoning. For example, the input of AND3 can only be 2, 5 or 6 according to the pattern table in Figure 2. Then, all of them are considered for gate F'. Since pattern 5 has already been connected to F', we can exclude the possibility that it is connected to the middle input pin of F'. Likewise, the pattern index for a layout gate may have multiple candidate possibilities due to the missing connection information. In Figure 3, the inverter E' has 3 possible input connections "0/1/3", which lead to three possible pattern indices 4, 5 and 6. From the pattern table in Figure 2(b), we know 4 cannot be an input to AND3, and therefore we can remove 4. Since pattern 5 has already been input of F' from C', E' cannot be 5, and thus we can identify the pattern index for E' as 6. This identification also tells that the middle input to F' must be from pattern index 2, i.e., gate B'.

3.3 Pruning by Hints from Design Conventions

Due to the missing BEOL information, the attack must guess and keep a set of candidate solutions. Even after the simple pruning described in Section 3.2, there could still be multiple candidate pattern indices for a layout gate. For example, in Figure 3, layout gate G' can be matched with either index 1 or index 8. We use the hints from design conventions as below to perform further pruning.

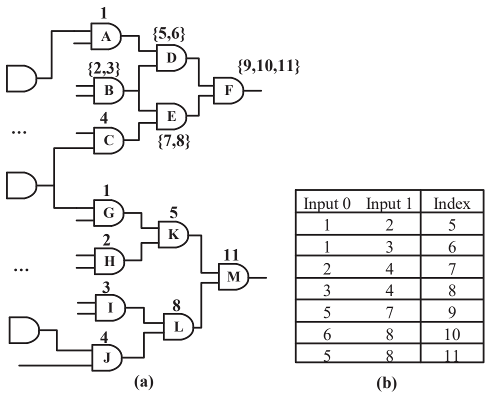
(1) **Load capacitance constraint**: To ensure the signal integrity, a gate in the technology library should honor a limited load capacitance on its fanouts. When we are examining a potential connection in BEOL layers, those that violate the load capacitance constraint can be excluded from being a candidate.
(2) **Timing constraint**: Digital circuit is sensitive to setup/hold time constraint. We can obtain an estimate to the actual arrival time and required time on each node of the path through an educated guess on clock period. If a candidate connection violates the timing constraints, it should be excluded from possible BEOL connections in matching cells.
(3) **Directionality of the dangling wires**: Although BEOL connections are hidden, the directionality of dangling wires at lower FEOL layers can still suggest the direction for reconnection. For example, if a source gate has a dangling wire pointing toward the south of the layout, those candidates of sink gates located in its north are most likely disregarded.

If there are still multiple candidates after all pruning, we choose the one with minimum BEOL wirelength.

3.4 Propagating Candidates along Subcircuit

When multiple candidate indices of a layout gate cannot be immediately pruned to a single one, they are propagated toward circuit outputs. When a candidate is combined with a fanout gate, and the combined pattern cannot be found in the netlist pattern tables, this candidate can be pruned out.

Since the topology of a subcircuit is generally a DAG (Directed Acyclic Graph), the candidate propagation on it faces the history consistency issue. Please look at the example in Figure 4, which has two candidates 2 and 3 for gate B. When they are propagated to gate D, we have two candidates, 5 that results from combining 1 of gate A with 2, and 6 that is from combining 1 with 3. Likewise, two candidates 7 and 8 are obtained at gate E. When the candidates from D and E are propagated to gate F, there are four combinations. Some of the combinations are illegal. For example, 11 that combines 5 from D and 8 from E is illegal, because this combination implies that B must be both 2 and 3 simultaneously. One can track the whole propagation history, but such tracking would cost either huge runtime or memory storage. Hence, we choose not to track the history with the risk of keep illegal candidate. However, we can



Input 0	Input 1	Index
1	2	5
1	3	6
2	4	7
3	4	8
5	7	9
6	8	10
5	8	11

Figure 4: Example of propagating candidates along subcircuit: (a) the subcircuit from layout and (b) the pattern table from netlist.

show that the probability of finally keeping the illegal candidate is very low. If the pattern $(AND2, 5, 8)$ does not exist in the netlist pattern table, the combination of 5 and 8 on AND gate can be directly pruned. Otherwise, like in Figure 4(a), $(AND2, 5, 8)$ is a legal pattern and exists in the pattern table. Such probability that a legal pattern has the same pattern table as an illegal candidate is very low. This probability can be further reduced by matching tree-only patterns first.

3.5 Matching of the Entire Circuit

There are two different ways of recognizing the layout of an entire circuit. One is to build a single pattern table of the entire netlist and use it to match the layout. Its drawback is that the table can be either too huge or very slow to search. Thus, we divide the netlist into subcircuits and use relatively small pattern tables of these subcircuits to match the layout. Sometimes, there is no need to recognize the entire layout. For instance, an attacker plans to insert a Trojan into encryption circuit. Then, only the encryption part of the layout needs to be recognized.

Given a list of subcircuits with pattern tables, we match each of them one by one on the layout. Once some layout gates are recognized, we use the pattern tables of the fanout of the recognized gates for further matching. If the complete netlist is available, I/O pins can be treated as matched. As such, the matching starts from primary inputs and moves toward primary output in topological order. The matching terminates when no more layout gate can be further recognized.

4 EXPERIMENT RESULTS

4.1 Experiment Setup

We evaluate our technique by using ISCAS'85 and ITC'99 benchmark suites. All the designs are synthesized by Synopsys Design Compiler using 45nm technology library. Placement and routing are implemented by Cadence SoC Encounter. The bijective mapping attack is solved by an off-the-shelf SAT solver isat3 [5]. The structural pattern matching attack is implemented with C language. The experiments run at Intel Xeon CPU with 2.8GHz frequency and CentOS Linux operating system. We evaluate the effectiveness of the attack methods by identifying the matching ratio that shows the percentage of correctly matched cells. The testcases are all simulated to be split manufactured, and then further prepared in two ways, one without k-security defense and the other with k-security defense [3]. To limit the overhead, k-security is applied to a small portion (about 5 or10 percent) of the circuits with defense.

Table 3: Experiment results on cases without k-security defense ("MR" indicates the ratio of correctly matched cell).

Design	#cells	Split layer	SAT bijective mapping		Structural pattern matching	
			MR(%)	Runtime	MR(%)	Runtime
c432	109	M3	100.00	0.9s	100.00	0.2s
c1355	222	M3	100.00	5.1s	100.00	0.9s
c1908	197	M3	100.00	3.2s	100.00	0.4s
c2670	374	M3	95.72	28.7s	97.06	14.2s
c3540	588	M3	100.00	56.9s	100.00	1m37.3s
c5315	819	M4	100.00	4m1.8s	100.00	21.9s
c6288	1889	M3	-	>48h	100.00	10.1s
c7552	834	M3	100.00	2m51.4s	99.76	9.5s
c880	192	M3	100.00	2.5s	100.00	0.4s
b07	258	M3	100.00	10.7s	100.00	5.8s
b11	345	M3	100.00	13.0s	100.00	1.9s
b13	175	M3	100.00	12.9s	100.00	0.7s
b14	2743	M4	100.00	87m9.9s	100.00	2m34.5s
b15	5533	M5	-	>48h	100.00	164m16.4s
b17	17161	M6	-	>48h	98.51	1511m29.9s

4.2 Experiments on Cases without k-security Defense

Table 3 shows the experimental results on cases with k-security defense. Although our structural pattern matching algorithm is a heuristic, it can reach the matching ratio similar to the SAT-based bijective mapping attack and achieve 100% matching ratio in many cases. In only one design (c7552), the structural pattern matching has slightly lower matching ratio because of its heuristic nature. By incorporating the hints from design conventions, the structural pattern matching sometimes (c2670) provides more accurate attack than bijective mapping. The runtime advantage of the structural pattern matching is obvious in larger cases, and it is usually several times faster than bijective mapping. In a few big cases, the bijective mapping could not finish after 48 hours, because the SAT instance contains huge variables and constraints.

4.3 Experiments on Cases with k-security Defense

We implemented the benchmarks with a certain portion (5% or 10%) of cells being protected by k-security. Each design has 6 different secured layouts: 5/10% cells in 2-security, 5/10% cells in 3-security and 5/10% cells in 4-security. Both the matching ratio (MR) and runtime of attacks are shown in Table 4.

Table 4: Experiment results on cases with k-security defense ("MR" indicates the ratio of correctly matched cell). Number in bold indicates higher ratio than the expected success rate that k-security can guarantee.

Design	Defense	#cell	SAT bijective mapping		Structural pattern matching	
	type	secured	MR(%)	Runtime	MR(%)	Runtime
c432	2-secure	5%	**98.17**	0.9s	**100.00**	0.3s
		10%	**96.33**	3.1s	**96.33**	0.6s
	3-secure	5%	**98.17**	1.3s	**98.17**	1.8s
		10%	87.16	1.0s	**94.50**	5.9s
	4-secure	5%	92.66	0.9s	**97.25**	0.8s
		10%	90.83	1.2s	**98.17**	1.3s
c1355	2-secure	5%	94.59	6.6s	**98.65**	1.0s
		10%	91.89	5.9s	**95.95**	3.0s
	3-secure	5%	92.34	5.6s	**97.75**	1.8s
		10%	**95.50**	5.2s	**96.85**	2.1s
	4-secure	5%	95.50	5.5s	**97.30**	5.8s
		10%	89.19	5.4s	90.54	19.0s
c2670	2-secure	5%	96.79	37.1s	**98.89**	8.0s
		10%	91.44	36.8s	**96.26**	16.5s
	3-secure	5%	**97.33**	32.1s	**98.66**	10.6s
		10%	91.44	33.5s	**93.58**	21.5s
	4-secure	5%	94.65	37.7s	95.72	10.2s
		10%	90.91	36.2s	**93.32**	47.0s
c3540	2-secure	5%	**98.30**	58.1s	**99.49**	2m16.9s
		10%	**96.94**	55.2s	**98.13**	11m29.3s
	3-secure	5%	**98.64**	55.9s	**97.62**	58.6s
		10%	**96.43**	57.8s	**93.54**	9m47.6s
	4-secure	5%	**100.00**	1m10.2s	**97.45**	1m25.8s
		10%	**94.05**	1m0.2s	92.18	17m12.2s
c5315	2-secure	5%	95.73	4m3.8s	**97.68**	1m11.2s
		10%	91.70	3m48.6s	**95.73**	3m28.8s
	3-secure	5%	96.34	3m59.5s	**97.56**	54.2s
		10%	93.28	3m51.4s	**94.51**	2m18.5s
	4-secure	5%	95.60	3m44.5s	**97.56**	1m3.4s
		10%	90.35	4m19.1s	91.82	4m0.2s
Average	2-secure	5%	96.72	1m9.3s	**98.94**	43.5s
		10%	93.66	1m5.9s	**96.48**	3m3.64s
	3-secure	5%	96.56	1m6.9s	**97.95**	25.4s
		10%	92.76	1m5.8s	**94.60**	2m31.1s
	4-secure	5%	95.68	1m7.8s	**97.06**	33.2s
		10%	91.07	1m12.4s	**93.21**	4m27.9s

Our structural pattern matching attack mostly obtains higher matching ratio than the expected success rate that k-security defense can guarantee. Such results are boldface entries in the table. For example, benchmark c432 with 10% cells defended by 2-security is expected to have about 95% matching ratio under attack. This is because cells with k-security protection can be successfully recognized with the probability $1/k$. The experimental result shows that the theoretical guarantee of k-security cannot always be realized in practice, especially under our structural pattern matching attack, which significantly outperforms the SAT-based bijective mapping attack. On the other hand, the runtime advantage of structural pattern matching attack becomes less obvious compared with cases without k-security defense. Some cases show more runtime than the SAT-based bijective mapping attack, because we divide the netlist into subcircuits and use them to match the layout individually.

5 CONCLUSION

This paper considers the scenario where attackers have netlist information and attempt to recognize split manufactured layout for Trojan insertion, which was not fully discussed in existing works. We develop a new attack technique based on structural pattern matching to address the drawbacks of earlier mentioned SAT-based bijective mapping attack. Experiment results show that it is more efficient and more scalable than the bijective mapping attack. Unlike the theoretical metric of k-security, our proposed attack method provides an alternative way to evaluate the security of split manufactured ICs for Trojan insertion in practice.

ACKNOWLEDGMENTS

This work is partially supported by NSF (CNS-1618824), NSF (CNS-1618797), SRC (2016-TS-2688) and SRC (2016-TS-2689).

REFERENCES

[1] Y. Bi, J. Yuan, and Y. Jin. 2015. Beyond the Interconnections: Split Manufacturing in RF Designs. *MDPI Electronics* (2015), 541–564.

[2] S. Dupuis, P. S. Ba, G. Di Natale, M. L. Flottes, and B. Rouzeyre. 2014. A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans. *IEEE International On-Line Testing Symposium* (2014), 49–54.

[3] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara. 2013. Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. *USENIX Conference on Security* (2013), 495–510.

[4] Intelligence Advanced Research Projects Activity. 2011. Trusted Integrated Circuits Program. https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6.

[5] iSAT3. 2014. https://projects.avacs.org/projects/isat3.

[6] R. W. Jarvis and M. G. McIntyre. 2004. Split manufacturing method for advanced semiconductor circuits. *US Patent no. 7195931* (2004).

[7] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan. 2018. A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion. *ASPDAC* (2018), 265–270.

[8] J. MagaÃśa, D. Shi, J. Melchert, and A. Davoodi. 2017. Are Proximity Attacks a Threat to the Security of Split Manufacturing of Integrated Circuits? *TVLSI* (2017), 3406–3419.

[9] S. Mitra, H.-S.P. Wong, and S. Wong. 2015. Stopping Hardware Trojans in Their Tracks. http://spectrum.ieee.org/semiconductors/design/stopping-hardware-trojans-in-their-tracks.

[10] C.T.O. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar. 2015. Automatic obfuscated cell layout for trusted split-foundry design. *HOST* (2015), 56–61.

[11] S. Patnaik, J. Knechtel, M. Ashraf, and O. Sinanoglu. 2018. Concerted wire lifting: Enabling secure and cost-effective split manufacturing. *ASPDAC* (2018), 251–258.

[12] J. Rajendran, O. Sinanoglu, and R. Karri. 2013. Is split manufacturing secure? *DATE* (2013), 1259–1264.

[13] A. Sengupta, S. Patnaik, J. Knechtel, M. Ashraf, S. Garg, and O. Sinanoglu. 2017. Rethinking split manufacturing: An information-theoretic approach with secure layout techniques. *ICCAD* (2017), 329–326.

[14] M. Tehranipoor and F. Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design Test of Computers* (2010), 10–25.

[15] K. Vaidyanathan, B. P Das, E. Sumbul, R. Liu, and L. Pileggi. 2014. Building Trusted ICs using Split Fabrication. *HOST* (2014), 1–6.

[16] K. Vaidyanathan, B. P Das, E. Sumbul, R. Liu, and L. Pileggi. 2014. Detecting Reliability Attacks During Split Fabrication Using Test-only BEOL Stack. *DAC* (2014), 156:1–156:6.

[17] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi. 2014. Efficient and Secure Intellectual Property (IP) Design for Split Fabrication. *HOST* (2014), 13–18.

[18] J. Valamehr, T. Sherwood, R. Kastner, D. Marangoni-Simonsen, T. Huffmire, C. Irvine, and T. Levin. 2013. A 3-D Split Manufacturing Approach to Trustworthy System Development. *TCAD* (2013), 611–615.

[19] Y. Wang, P. Chen, J. Hu, and J. Rajendran. 2016. The cat and mouse in split manufacturing. *DAC* (2016), 1–6.

[20] Y. Wang, P. Chen, J. Hu, and J. Rajendran. 2017. Routing perturbation for enhanced security in split manufacturing. *ASPDAC* (2017), 605–510.

[21] Y. Xie, C. Bao, and A. Srivastava. 2017. Security-Aware 2.5D Integrated Circuit Design Flow Against Hardware IP Piracy. *Computer* (2017), 62–71.

[22] M. Zhao and S. S. Sapatnekar. 2001. A new structural pattern matching algorithm for technology mapping. *DAC* (2001), 371–376.