Circuit Obfuscation and Oracle-guided Attacks: Who can Prevail?

Kaveh Shamsi[1], Meng Li[2], Travis Meade[1], Zheng Zhao[2], David Z. Pan[2], and Yier Jin[1]
[1]ECE Department, University of Central Florida, Orlando, FL, USA
[2]ECE Department, University of Texas at Austin, Austin, TX, USA
[1]{kaveh, travm12}@knights.ucf.edu, [2]{meng_li, zzhao, dpan}@utexas.edu,
[1]yier.jin@eecs.ucf.edu

ABSTRACT

This paper provides a systematization of knowledge in the domain of integrated circuit protection through obfuscation with a focus on the recent Boolean satisfiability (SAT) attacks. The study systematically combines real-world IC reverse engineering reports, experimental results using the most recent oracle-guided attacks, and concepts in machine-learning and cryptography to draw a map of the state-of-the-art of IC obfuscation and future challenges and opportunities.

1. INTRODUCTION

Two of the major security and privacy concerns in today's integrated circuit (IC) supply chain are 1) The threat of integrated circuit reverse engineering (RE), 2) the threat of malicious hardware modifications known as hardware Trojans. IC reverse engineering technologies have been constantly improving over the past decades resulting in a near fully automated process for extracting various kinds of information from modern ICs fabricated with nanometer scale features [39, 25, 1]. The trend of globalization of IC manufacturing has also continued, leaving designers in US with few competitive trusted fabrication facilities. Fabless design houses carry out high-profit design and intellectual property development tasks while the complex and costly manufacturing services are outsourced to off-shored foundries [33].

Over the past couple of decades, various design-time techniques have been proposed for thwarting the above threats [29]. The most prominent of this techniques include: 1) IC camouflaging: based on creating indistinguishable layout structures to hinder reverse engineering by end-users; 2) Logic locking: to lock down a circuit with active key bits that are programmed post-manufacturing hiding away details from the foundry as well as the end-user; 3) Split-manufacturing: manufacturing only part of the IC (e.g. Front End Of Line layers) in the untrusted fab. The focus of this paper is on the first two techniques and their security. Both of IC camouflaging and logic locking can be classified as circuit "obfuscation" schemes as they modify the design to obscure information away from the attacker while maintaining its functionality.

The two most important question regarding circuit obfuscation is whether they are secure and whether they can be implemented with low overhead. In 2015 a Boolean Satisfiability (SAT) based attack [17, 38] was proposed that was shown to be able to deobfuscate a significant portion of existing low-overhead IC camouflaging and logic locking schemes. Since then the problem of logic obfuscation/deobfuscation has reemerged. In this paper we will take a systematic look at some of the most prominent circuit obfuscation schemes discussed in literature with respect to the SAT attack and beyond the SAT attack. The main contribution of the paper is that it summarizes a broad range of adversary-models, recent attacks, most resilient low-overhead obfuscation schemes, and the silicon level technologies that enable them. This can help researchers identify the open problems in the domain.

The paper is organized as follows. We begin by first surveying IC camouflaging and logic locking algorithms and their nano-device primitives presented in prior work in Section 2. We then discuss the various threats posed by end-users and the foundry and the emergence of SAT attacks in Section 3. We evaluate the security of select circuit obfuscation schemes under these threats in Section 4. Future directions are discussed in Section 5. Section 6 concludes the paper.

2. BACKGROUND

2.1 Integrated Circuit Supply Chain

The modern semiconductor industry operates on a global scale with a distributed design, verification and fabrication network [33]. The fabless design model has shown to be productive for several industries and it is predicted that it will continue to grow [33]. In this model, the design and the verification are performed in the design house. Then the design layout in form of a GDSII file is sent to the foundry. Fabrication, and possibly testing and packaging is performed at the foundry and the devices are shipped back to the designer for retail. The foundry has a special position in the IC supply chain. It has possession of the layout of the device, therefore, all hard silicon features are known to the foundry and it has the ability to maliciously modify the design.

Once the ICs arrive at the end-user's disposal, the threat of physical reverse engineering begins. There are standard techniques and automatic tools for an end-user to remove the IC package, scan the IC layer by layer, image each layer and then use image-recognition techniques to reconstruct the layout and subsequently the netlist of the chip [40]. Furthermore, micro-probing technology allows the attacker to read values stored in Flash or other active memories by probing specific wires that carry the information [25].

2.2 IC Camouflaging

IC camouflaging is the idea of fabricating integrated circuits with layout structures that are difficult to resolve to a certain functionality. The building blocks of IC camouflaging are silicon structures for which their functionality is difficult to determine through re-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada

© 2017 ACM. ISBN 978-1-4503-4972-7/17/05…5.00

DOI: http://dx.doi.org/10.1145/3060403.3060494

verse engineering with microscopy, or chemical and mechanical processes [15]. Hence, designing these building blocks is a nano-device fabrication problem. There is an array of such technologies available [4, 11, 14, 10, 5] most of which are compatible with the CMOS process. Since our paper's focus is on the Boolean function level we broadly classify all camouflaging devices into the following categories of primitives:

- **Connectors**: these include metal-to-metal [10], or metal-to-diffusion [12] connectors that are either connected that appear unconnected, or unconnected that appear connected.
- **Fixed Transistors**: which are transistors that are always-on or always-off regardless of the gate-terminal voltage [5, 6, 4].

There are various nano-device structures that are used to implement each of the above primitives. A few that are important for the purpose of this paper include: 1) Doping-based camouflaging where two transistors with similar gate and contact structures but different doping types behave differently creating fixed transistors or connector diodes [36, 36]; 2) Metal-to-metal vias with a middle-gap [10]; 3) Mg/MgO metal-to-metal vias [9]; 4) Metal-to-diffusion connectors with a thin insulating layer between the drain and the contact arriving on top of it [12]; 5) Light-density-doping (LDD) can also be used to render always-on or always-off transistors [4].

Larger camouflaged functional units can be built using these primitives: 1) camouflaged gates that implement different functionalities using camouflaged connectors [28], 2) a single camouflaged bit using a voltage divider with two camouflaged contacts or a fixed transistor, 3) gates with dummy inputs using fixed transistors [21], 4) dummy paths using a wire and two fake metal-to-metal vias at the end-points can be built. These unites can then be dispersed throughout the netlist with different strategies [28] which are discussed further in Section 4.

2.3 Logic Locking

While the above camouflaging technologies make it difficult for an end-user to reverse engineer the correct netlist from the layout, the true functionality of the camouflaging units, such as which vias are fake or true is still disclosed to the foundry. In order to prevent the foundry from learning design secrets the secret should not be part of the layout.

Logic locking is an obfuscation scheme that relies on active key-inputs that are programmed on the chip after the chip returns from fabrication effectively hiding them from the foundry [31]. The goal is to have the circuit produce incorrect outputs for incorrect key values. At the silicon level, logic locking can be implemented either with a dedicated tamper-proof memory feeding key values, or with programmable primitives dispersed throughout the netlist. Some possible programmable primitives are oxide-breakdown-based anti-fuse devices, emerging non-volatile memory devices such as RRAMs [43] and other non-volatile programmable devices whose state is difficult to determine from microscopy.

At the functional level there are various strategies for logic locking. We can broadly categorize them into sequential logic locking and combinational logic locking. Combinational logic locking schemes corrupt the output of a directed-acyclic-graph (DAG) circuit using key-inputs. Notable schemes include adding key-controlled XOR/XNOR or MUX gates in randomly selected [31] or judiciously selected [30, 27] locations in the circuit; replacing gates with look-up-tables [7, 42] that store key-values; shuffling wires with key-controlled switch-boxes [32, 43]; and a few other gate insertion/replacement schemes [16].

Sequential logic locking schemes [8, 18] focus on adding dummy states to the state-transition-graph (STG) of a sequential circuit. We will focus on combinational obfuscation schemes in this paper as they assume a stronger threat-model which is that the attacker has access to all registers through the scan-chain. Furthermore, if a secure combinational obfuscation scheme exists, simply obfuscating the state-generation function of a sequential circuit with that scheme will result in a secure sequential obfuscation.

Logic-locking can be extended to prevent overproduction by including a physical unclonable function (PUF) onto the chip applying a device-unique mapping to the key. With the PUF every fabricated chip will have a different key value.

2.4 Circuit Diversification

Assume a circuit is implemented with a Boolean function c_o there is a large set of Boolean circuits that implement the same function, $c_o \in C$. Circuit diversification schemes [26, 24] try to replace c_o with another circuit $c_d \in C$ such that it is difficult for an attacker to find the structures in c_o given c_d. For instance, by replacing pieces of logic with another randomly selected equivalent circuit can potentially disguise an attacker-known structure in the circuit preventing graph-isomorphism based component matching in the netlist. Note that by itself this does not hinder recovery of the netlist of c_d which is functionally equivalent to c_o.

3. ADVERSARIAL MODELS

We will describe a classification of threats accounting for realistic reverse engineering reports such as [25]. The supply chain in view of this categorization is depicted in Figure 1.

3.1 High-level Recognition

Typically the first step in IC reverse engineering is optically imaging the chip. With optical images although transistor details in advanced technologies are not visible, the boundaries of memory arrays, digital logic or analog blocks are recognizable. This allows the attacker to establish a high-level knowledge of the position of different components. Next the attacker can proceed with scanning electron microscopy (SEM) and imaging to get down to nanometer scale features of the chip.

In addition, the attacker can utilize micro-probing technologies to probe accessible portions of the circuit. In order to probe the circuit the adversary has to find the particular wires that she is interested in such as the bus that carries the decrypted values from Flash memory to the processor. It is intuitive that with some knowledge of the high-level functionality and knowledge of the location of important wires the attacker can probe accessible wires or attach hardware Trojans to them for the case of a foundry adversary. Therefore, it is safer to assume that an obfuscation scheme will prevent hardware Trojan insertion only if it prevents high-level recognition which as we discuss later can be rather difficult to prevent using low-overhead obfuscation.

3.2 Netlist Recovery

A more difficult task for the adversary is to extract an unambiguous circuit netlist from the IC. In this case if the circuit is camouflaged or locked the attacker will recover an obfuscated netlist as seen in Figure 1. This obfuscated netlist can have camouflaged components that can implement a set of possible functions, or the netlist has a set of active key inputs that are unknown. Both cases can be modeled as Boolean function $c_e : I \times K \rightarrow O$ where I and K are input and key space respectively, and there exists a correct key k_* such that $\forall i \in I,\ c_e(i, k) = c_o(i)$ where c_o is the unobfuscated circuit. In terms of deobfuscation accuracy there are two cases:

- **Exact Attack**: the attacker wants to exactly find the original netlist or a functional equivalent of it.
- **Approximate Attack**: the attacker can allow for some inaccuracy in the attack.

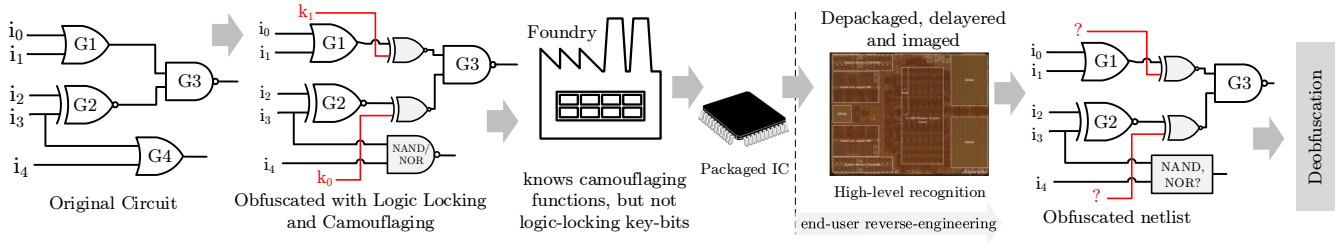

Figure 1: Integrated circuit life-cycle through obfuscation and deobfuscation (The chip image is from [1]).

Most recent and strong deobfuscation attacks assume that the attacker also has access to an "oracle", an unlocked or functional circuit that returns correct outputs for chosen inputs. In this dimension the attacks can be categorized as:

- **oracle-guided**: the attacker can query the oracle circuit with i and receive the output $c_o(i)$.
- **obfuscated-oracle-guided**: the attacker can query the oracle with i and will receive $\gamma(c_o(\alpha(i)))$ where α and γ are potentially obfuscated functions.
- **oracle-less**: the attacker does not have access to an oracle.

Another significant aspects in evaluating the resiliency of logic obfuscation schemes is assumptions on the physical reverse engineering capabilities of the attacker. Attacks can be categorized in this dimension by considering which of the following capabilities the attacker has:

- the attacker can differentiate between doping types;
- attacker sees all connections as potentially-fake;
- attacker can differentiate between potentially-fake and true connections;
- attacker can differentiate between fake and true connections.

Based on the abstract camouflaging primitive categorization in Section 2.2 the above questions should cover the entire space of attacker physical reverse engineering capabilities. To stress the importance of these assumptions, we note that with the last assumption none of existing camouflaging schemes would be secure, and with the second assumption the simplest camouflaging schemes would be extremely difficult to deobfuscate. Currently we know that doping types are easily visible. There is no extensive documented efforts on breaking non-doping camouflaged connectors.

3.3 Evolution of Oracle-guided Attacks

The first works on logic locking proved the security of their schemes by assuming that in order for the adversary to recover the key the attacker has to solve the following equivalence equation $c_e(i,k) = c_o(i)$ which is NP-complete due to alternating quantifies and hence difficult to solve for k. However, the reasoning is not accurate because the Boolean equation for c_o is not available to the attacker. The oracle-guided threat model then arrived in the work of Rajendran et al. [28] where they proposed that by sensitizing the output of the circuit to a particular key-bit that key-bit can be revealed through querying.

The SAT attack [17, 37, 34] which is the most recent and strongest oracle-guided attack queries the circuit on input patterns that result in disagreements for different key values. A novel aspect of the SAT attack is that the circuit is modeled as a SAT formula and the input-output observations are also stored as Boolean conjunctive clauses in the SAT formula. This allows an efficient SAT solver to wrestle with the NP-complete problem to both find disagreeing queries and find a key value that satisfies all of the observed queries with high performance. The SAT attack shows great capability against most known logic locking schemes and camouflaging schemes that replace a portion of gates in the design with camouflaged gates.

3.4 A General Model for Oracle-guided Deobfuscation

It is possible to construct a set of general security criteria for resiliency against deobfuscation. As was pointed out in [21, 34] oracle-guided attacks against circuit obfuscation are equivalent to learning a function with samples. The attacker begins with a set of possible functionalities in a function space C. This set C essentially denotes all the information that the attacker begins with including the obfuscated netlist (without the obfuscated netlist itself as a hint, C will be the set of all possible circuits). Then the attacker can query the oracle and use the observed information to further prune C to get to the target function c_o.

Given this model, an obfuscation scheme is computationally secure if there is no algorithm that will find c_o in time polynomial in the circuit size. An obfuscation is information-theoretically secure if there is no algorithm that finds c_o any better than random guessing given infinite amount of time. In machine learning terms approximation can also be modeled using the probably-approximately-correct (PAC)-learning scheme [2, 20]. In PAC-learning terms, an obfuscation is computationally approximation resilient if there exists no polynomial time randomized algorithm A that can learn an ϵ-approximation of the function c_o with success rate of $1-\delta$ where δ and ϵ are arbitrarily small.

4. CASE STUDY DEFENSES

4.1 Programmable Logic Blocks

Using programmable logic blocks have been suggested in literature as an obfuscation strategy [22, 42]. Various forms of programmable logic such as SRAM-LUT based FPGAs, Anti-fuse FPGA, One-time Programmable PLAs, LUTs with doping based camouflaged configuration bits [36], LUTs with dummy via camouflaging, etc. can be used as an obfuscated circuit wherein the configuration bits are the secret key. If the attacker cannot observe the configuration bits the device appears to implement a large number of possible functionalities. Against high-level recognition, the layout of programmable device fabrics has a repetitive structure so that unlike an ASIC the optical images do not reveal logic boundaries. Essentially the circuit functionality is programmed electrically rather than physically as in ASICs.

It is easy to show that programmable logic based obfuscation is secure against key recovery if the input size is large. A programmable logic fabric that has g gates and reconfigurable interconnects, will implement all possible functions that can be implemented by rearranging l gates if $l \ll g$. Due to the high expressiveness of the programmable logic each query will reveal the output of the logic for only that particular input. Against oracle-guided approximation, the rate at which any algorithm learns the truth-table of the programmable block is linear in the size of the truth-table and thus for a large enough input size this rate can be slow.

The size of the programmable logic plays an important role. This is the main reason why replacing small gates with small LUTs [42] does not provide exponential security against SAT attacks. We ob-

fuscated the c432 ISCAS benchmark circuit by replacing gates with LUTs of various sizes and attacked it using our C++ SAT attack framework[1]. As can be seen from Table 4.1 the number of necessary queries and running time in seconds do not increase exponentially with the number of small LUTs.

#LUT-inputs	2		3		5		6	
#LUTs	time	#iter	time	#iter	time	#iter	time	#iter
4	0.02	8	0.04	24	0.18	68	0.72	171
8	0.03	20	0.12	62	0.53	132	2.2	234
16	0.15	41	0.39	95	1.53	183	3.12	282
20	0.18	46	1.03	156	1.97	205	3.37	276
32	0.58	81	1.82	152	2.06	205	2.58	257

Table 1: LUT insertion resiliency with respect to the number of inputs to the LUT and the number of LUTs inserted.

While programmable units can provide possibly the maximum level of security available from an obfuscation scheme, they come at high area, power and delay costs. Averaged across different functions, FPGAs can cost 30X more area, 3-4X more delay, and 10X more power consumption [19]. Therefore, Liu et al. [23] proposed replacing only critical modules in the design such as the instruction decoder, in the case of a SPARC processor with FPGA fabric which can reduce the overall overhead costs.

4.2 Point-function based Logic Locking and IC Camouflaging

Since the advent of the SAT attacks several works have targeted defeating the attack by utilizing special key functions that are hard to break given oracle samples [21, 44, 45, 47]. A point-function $P(i, k)$ will return 0 for all $i \neq k$ and 1 otherwise. This function can be implemented by XORing bits in the k and i vectors and then feeding the resulting bits into an AND-tree. For IC camouflaging, small BUF/INV camouflaging gates can be used at the input of the AND-tree and for logic locking programmable switches can be used. It is easy to see that for this function querying i values will not reveal k unless $i = k$ which has a low probability if the bit-length of i and k is large.

This idea of low activity key functions is the basis of all of the proposed point-function schemes. Since the output corruptibility of such schemes is low they have to be combined with other high-corruptibility schemes such as random XOR/XNOR logic locking or gate-replacement IC camouflaging.

While it is easy to see that all these schemes provide computational security against exact SAT attacks for large input sizes, they fail to resist an approximation attack. This was exploited in the work of Shamsi et al. [34] where an approximate SAT attack was developed that stops the querying process early and justifies the entire key vector. The returned key will have effectively solved the high-corruptibility portions of the obfuscation. Thus only the low-corruptibility obfuscation would remain which due to the fact that it is corrupting the output for only a single input pattern, the attacker has obtained a very good approximation of the original circuit in polynomial queries. Hence point-function obfuscation schemes lack resiliency against approximate oracle-guided attacks.

In addition, some of the point-function schemes are vulnerable to structural and functional analysis [46]. In such attacks the attacker performs a structural search in the circuit to find the point-function block, and since most these schemes insert the point-function into the circuit at a single-point removing it from the obfuscated circuit will return the original circuit.

[1]All tests in this paper were run on a quad-core Intel Xeon E3 processor with a 3.4GHz CPU, and 16GB memory.

Against high-level recognition, as stand-alone solutions these schemes provide no security, since they have one point of entry in the circuit so only a single wire is affected by the obfuscation and the structure of the original circuit is largely preserved.

4.3 Layout Flooding

Since the SAT attack was proposed most researchers in academia have shifted towards developing new obfuscation schemes that can withstand the attack. For IC camouflaging the documented attacks [17] were against gate-level IC camouflaging where randomly selected or clique selected gates in the circuit are replaced with camouflaged gates that implement a possible set of three or four functionalities.

A particular camouflaging scheme that was not analyzed under the SAT attacks is present in a series of industrial patents and is currently part of commercialized circuit camouflaging technologies [13]. This scheme is based on flooding the layout of the circuit with dummy gates and dummy wires and then connecting them to active logic as seen in Figure 2.

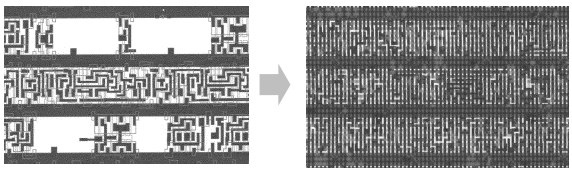

Figure 2: Layout flooding from [13].

While adding a large number of dummy elements to the design and filling empty areas of the chip with dummy logic can provide difficulties during layout image recognition, the specific insertion and connection algorithm determines SAT attack resiliency. The algorithms discussed in [13] insert filler cells into the empty areas of the layout and then connect them to active logic. However, the connections are passive with respect to the original circuit, meaning that the output of filler cells is not disrupting the functionality of the original circuit. Therefore, other than hindering cell recognition during layer-by-layer image analysis the defense does not affect the SAT attack complexity.

Acyclic wire-flooding obfuscation scheme experiment. Along the direction of layout flooding, we designed a dummy wire flooding obfuscation scheme. In this scheme for a certain percentage of the wires in the circuit we connect them to a number of randomly selected location using a MUX effectively creating dummy paths with the condition that no logical loops are created. We applied this scheme to a set of randomly synthesized circuits of input-size 8 with a few 100 gates and report the results in Table 4.3. As can be seen once the circuit is flooded with dummy paths the deobfuscation process becomes difficult. Around 50% coverage with more than 8-input MUXs the attack becomes ineffective. However, the overhead of adding all these dummy paths to 50% of the wires in the circuit can be high.

%wires	10%		20%		30%		50%	
MUX-size	time	#iter	time	#iter	time	#iter	time	#iter
2	0.06	10	0.29	17	0.54	28	2.22	38
4	0.29	19	1.02	22	2.89	41	27.3	68
6	0.48	20	1.91	25	10.1	47	1286	84
8	0.61	17	5.26	37	56	56	TO	TO

Table 2: acyclic wire-flooding obfuscation scheme.

Cyclic wire-flooding obfuscation. Shamsi et al. [35] recently proposed a topological obfuscation scheme that uses camouflaged/-programmable connectors to insert minimum number of dummy

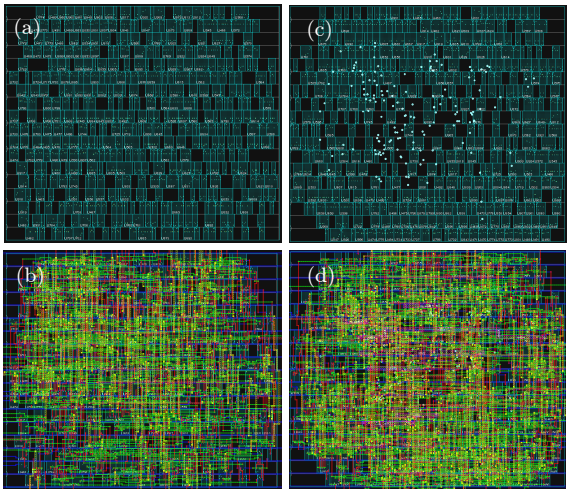

Figure 3: Cyclic camouflaging on the c1908 ISCAS benchmark where (a) and (b) are the original circuit, (c) shows the dispersed camouflaging vias and (d) is the routed obfuscated circuit. In Layout flooding schemes only wire overhead is incurred.

wires into the logic to create "unresolvable loops". If logical loops are created in the circuit the SAT attack will not be able to proceed since the obfuscated circuit can no longer be represented as a DAG. Hence combining this idea with layout flooding schemes, such that not only are all empty areas filled with dummy structures, but that many unresolvable loops are also created can result in a low overhead highly resilient obfuscation for ASICs. The looped structure of the circuit graph adds an additional dimension to the complexity of the SAT attack and therefore should result in higher attack complexities than acyclic flooding of the same size.

As for high-level recognition layout flooding schemes can potentially create security. The dummy connections can make it difficult for the attacker to find and trace specific wires or perform microsurgery to reach and probe arbitrary signals. Note that none of the obfuscation schemes discussed thus far are resilient against arbitrary probing. A layout implementation of the cyclic obfuscation scheme on the c1908 ISCAS benchmark can be seen in Figure 3.

4.4 Using Cryptographic Functions

Almost all IC camouflaging and logic locking schemes would break if the attacker can observe a significant number of internal signals. One of the most intriguing questions in cryptography over the last few decades has been to see whether there exists a general algorithm for obfuscating a function that remains secure even if all internal variables are public [3]. Although it is still an open problem, certain cryptographic primitives can be used to obfuscated specific functions providing security against probing. One such example which can be useful in authentication is the hashed password checking function. In this scheme a cryptographic hash of a secret key is stored on the device and then a comparison with this key is performed by taking the hash of the input and comparing it to the stored value. In such a circuit reading internal values will not reveal the password in plain-text.

It seems intuitive to utilize the numerous cryptographic blocks in obfuscation schemes. For instance the hash password checker can be used to raise a wire to signal the circuit to begin operation. Yasin et al. [48] also proposed inserting an AES block in the path of the secret key before feeding the key bits to key gates in logic locking. The security of the AES cipher against chosen plaintext attacks would imply the security of such an obfuscation against oracle-guided key-recovery. However, the main disadvantage of using cryptographic blocks in this way is that they can be vulnerable to structural/functional search attacks. For instance, the single wire that enables the circuit operation can be found and excluded from the circuit returning the original circuit. The AES block can also be found and simply removed from the circuit. Hiding these large blocks within the circuit with strong security guarantees would be very difficult and costly if not impossible.

5. FUTURE DIRECTIONS

5.1 Machine-learning Approaches

The existing SAT attacks are essentially machine-learning algorithms that learn the original functionality of the obfuscated circuit given input-output samples. However, machine-learning approaches can be applied to attack other aspects of obfuscation schemes. The obfuscation algorithm is public to the attacker and the synthesis and place & route tools that are used are a source of statistical signatures that can be detected through machine-learning. Generally, a machine-learning problem can be formed where the obfuscation algorithm, f was applied to the circuit and given samples of f the attacker wants to learn f^{-1} to reverse the obfuscation. Furthermore, machine-learning schemes can be used for searching structure or functionality of the circuit for finding obfuscation blocks such as point-functions and remove them from the circuit.

The reason for any machine-learning scheme to prevail in deobfuscation is that there exists small statistical variations in any obfuscation scheme that are dependant on the original circuit functionality. Unlike cryptography where these small variations are studied in depth and exploited in attacks, for circuit obfuscation this task has lacked motivation since few such attack have been documented. This direction also motivates designing algorithms that will destroy such secret-dependant variations. Circuit diversification schemes may be useful for this purpose.

5.2 Side-channel Attacks

The obfuscated circuit is physically implemented on a device. Therefore, they are vulnerable to power, timing, and electromagnetic side-channel attacks. Side-channels can reveal small statistical variations that are dependant on the key value. Thus far, most proposals have simply suggested that side-channel attacks will not be possible since the camouflaging elements that are used do not have secret-dependant power consumption profiles. However, there are various power models and higher order attacks that can be used. Therefore, designing such attacks and then preventing them with circuit and algorithmic solutions is a critical to the overall security.

5.3 Physical Aspects

The security of obfuscation schemes is significantly affected by the resiliency of the camouflaging primitives. For instance, if camouflaging metal-to-metal via can be made for which the attacker cannot differentiate between real and potentially-fake vias, it would be extremely difficult to attack layout flooding and cyclic interconnection camouflaging schemes. The ability of the attacker to tell apart real silicon constructs from potentially camouflaged ones allows the attacker to localize the uncertainty in the netlist to a fraction of the entire design. This is key to the success of SAT attacks on large circuits. This was also suggested in [41] where authors assert that resilient camouflaging primitives that can become ubiquitous in the netlist will create great attack complexity. Note that for logic locking it is more difficult to use ubiquitous programmable elements, due to the fact that each secret bit has to be programmed requiring programming circuitry that will help the attacker identify protected regions as well as creating additional overhead.

Another instance that stresses the importance of physical aspects of obfuscation is doping-based camouflaging. Various works [36] were published under the assumption that doping patterns cannot be detected with microscopy, however, there exists several microscopy and chemical processes to detect doping types. Unfortunately developing and securing camouflaging primitives is a nano-fabrication research problem and is outside the scope of the EDA community, however, it has a significant impact on the security analysis at the functional level.

6. CONCLUSION

In this paper the state-of-the-art in IC protection through obfuscation was evaluated. In summary it was demonstrated that the ubiquitous flooding of the circuit layout with camouflaging or programmable components and the utilization of cyclic circuit structures will substantially increase the complexity of reverse engineering attacks with low overhead. Further research into the design and resiliency testing of camouflaging nano-structures is essential to the the security of circuit obfuscation.

7. REFERENCES

[1] Chipwork. http://ww.chipworks.com/.

[2] J. L. Balcázar, J. Castro, D. Guijarro, J. Köbler, and W. Lindner. A general dimension for query learning. *Journal of Computer and System Sciences*, 73(6):924–940, 2007.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001.

[4] J. Baukus, L. Chow, and W. Clark. Permanently on transistor implemented using a double polysilicon layer cmos process with buried contact, May 25 2004. US Patent 6,740,942.

[5] J. Baukus, L. Chow, and W. Clark. Programmable connector/isolator and double polysilicon layer cmos process with buried contact using the same, May 17 2005. US Patent 6,893,916.

[6] J. P. Baukus, W. M. Clark Jr, L.-W. Chow, and A. R. Kramer. Integrated circuit security system and method with implanted interconnections, Feb. 2 1999. US Patent 5,866,933.

[7] A. C. Baumgarten. Preventing integrated circuit piracy using reconfigurable logic barriers. 2009.

[8] R. Chakraborty and S. Bhunia. HARPOON: An obfuscation-based soc design methodology for hardware protection. *IEEE J. Technol. Comput. Aided Design*, 28(10):1493–1502, 2009.

[9] S. Chen, J. Chen, D. Forte, J. Di, M. Tehranipoor, and L. Wang. Chip-level anti-reverse engineering using transformable interconnects. In *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, pages 109–114. IEEE, 2015.

[10] L. Chow, J. Baukus, and W. Clark. Integrated circuits protected against reverse engineering and method for fabricating the same using vias without metal terminations, Sept. 14 2004. US Patent 6,791,191.

[11] L. Chow, J. Baukus, and W. Clark. Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide, Nov. 13 2007. US Patent 7,294,935.

[12] L.-W. Chow, J. P. Baukus, and W. M. Clark Jr. Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide, Nov. 13 2007. US Patent 7,294,935.

[13] L. W. Chow, J. P. Baukus, B. J. Wang, and R. P. Cocchi. Camouflaging a standard cell based integrated circuit, Apr. 3 2012. US Patent 8,151,235.

[14] L.-W. Chow, W. M. Clark Jr, and J. P. Baukus. Covert transformation of transistor properties as a circuit protection method. US Patent 7,217,977.

[15] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang. Circuit camouflage integration for hardware ip protection. In *Proc. Design Automation Conf.*, pages 1–5. IEEE, 2014.

[16] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *Proc. IEEE Int. On-Line Testing Symposium*, pages 49–54. IEEE, 2014.

[17] M. El Massad, S. Garg, and M. V. Tripunitara. Integrated circuit (ic) decamouflaging: Reverse engineering camouflaged ics within minutes. In *NDSS*, 2016.

[18] F. Koushanfar. Provably secure active ic metering techniques for piracy avoidance and digital rights management. *IEEE Trans. on Information Forensics and Security*, 7(1):51–63, 2012.

[19] I. Kuon and J. Rose. Measuring the gap between fpgas and asics. 26(2):203–215, 2007.

[20] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proc. Int. Conf. on Machine Learning*, pages 148–156, 1994.

[21] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan. Provably secure camouflaging strategy for ic protection. In *ICCAD*, page 28. ACM, 2016.

[22] B. Liu and B. Wang. Embedded reconfigurable logic for asic design obfuscation against supply chain attacks. In *Proc. Design, Automation and Test in Eurpoe*, pages 1–6, 2014.

[23] B. Liu and B. Wang. Reconfiguration-based vlsi design for security. 5(1):98–108, 2015.

[24] J. T. McDonald, Y. Kim, and D. Koranek. Deterministic circuit variation for anti-tamper applications. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 68. ACM, 2011.

[25] T. Olivier and N. Dmitry. On the impact of automating the ic analysis process. In *Blackhat USA*, 2015.

[26] J. Parham, Y. Kim, et al. Hiding circuit components using boundary blurring techniques. In *Proc. of IEEE Annual Symposium on VLSI*, pages 5–7, 2010.

[27] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *DAC*, pages 83–89, 2012.

[28] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 709–720. ACM, 2013.

[29] J. Rajendran, O. Sinanoglu, and R. Karri. Regaining trust in vlsi design: Design-for-trust techniques. *Proceedings of the IEEE*, 102(8):1266–1282, 2014.

[30] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. Fault analysis-based logic encryption. *IEEE Trans. on Computers*, 64(2):410–424, 2015.

[31] J. A. Roy, F. Koushanfar, and I. L. Markov. Epic: Ending piracy of integrated circuits. In *Proc. Design, Automation and Test in Eurpoe*, DATE '08, pages 1069–1074, 2008.

[32] J. A. Roy, F. Koushanfar, and I. L. Markov. Protecting bus-based hardware ip by secret sharing. In *Proceedings of the 45th annual Design Automation Conference*, pages 846–851. ACM, 2008.

[33] S. K. Saha. Emerging business trends in the microelectronics industry. *Open Journal of Business and Management*, 4(01):105, 2015.

[34] K. Shamsi, M. Li, T. Meade, Z. Zhao, Y. Jin, and D. Z. Pan. Appsat: Approximately deobfuscating integrated circuits. In *Proc. IEEE Symp. Hardware-Oriented Security and Trust*. IEEE, 2017.

[35] K. Shamsi, M. Li, T. Meade, Z. Zhao, Y. Jin, and D. Z. Pan. Cyclic obfuscation for creating sat-unresolvable circuits. In *Proc. Great Lake Symp. on VLSI*. IEEE, 2017.

[36] M. Shiozaki, R. Hori, and T. Fujino. Diffusion programmable device: The device to prevent reverse engineering. *IACR Cryptology ePrint Archive*, 2014:109, 2014.

[37] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *Proc. IEEE Symp. Hardware-Oriented Security and Trust*, pages 137–143. IEEE, 2015.

[38] P. Subramanyan, N. Tsiskaridze, W. Li, A. Gascon, W. Y. Tan, A. Tiwari, N. Shankar, S. Seshia, and S. Malik. Reverse engineering digital circuits using structural and functional analyses. *IEEE Trans. on Emerging Topics in Computing*, 2(1):63–80, 2014.

[39] R. Torrance and D. James. The state-of-the-art in ic reverse engineering. In *Cryptographic Hardware and Embedded Systems*, pages 363–381. Springer, 2009.

[40] R. Torrance and D. James. The state-of-the-art in semiconductor reverse engineering. In *Proc. Design Automation Conf.* ACM, 2011.

[41] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu. Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques. *IEEE Transactions on Information Forensics and Security*, 12(1):64–77, 2017.

[42] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun. Hybrid stt-cmos designs for reverse-engineering prevention. In *Proc. Design Automation Conf.*, page 88. ACM, 2016.

[43] T. F. Wu, K. Ganesan, A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra. Tpad: Hardware trojan prevention and detection for trusted integrated circuits, 2015.

[44] Y. Xie and A. Srivastava. Mitigating sat attack on logic locking. http://eprint.iacr.org/2016/590.pdf.

[45] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu. Sarlock: Sat attack resistant logic locking. In *Proc. IEEE Symp. Hardware-Oriented Security and Trust*, pages 236–241, 2016.

[46] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. Security analysis of anti-sat. https://eprint.iacr.org/2016/896.pdf.

[47] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. Camoperturb: secure ic camouflaging for minterm protection. In *Proc. Int. Conf. on Computer Aided Design*, page 29. ACM, 2016.

[48] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. On improving the security of logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1411–1424, 2016.