Evolution of Logic Locking

Muhammad Yasin† and Ozgur Sinanoglu‡
{yasin, ozgursin}@nyu.edu
†Tandon School of Engineering, New York University, New York, USA
‡Division of Engineering, New York University Abu Dhabi, Abu Dhabi, U.A.E.

***Abstract*—The globalization of integrated circuit (IC) supply chain and the emergence of threats, such as intellectual property (IP) piracy, reverse engineering, and hardware Trojans, have forced semiconductor companies to revisit the trust in the supply chain. Logic locking is emerging as a popular and effective countermeasure against these threats. Over the years, multiple logic techniques have been developed. Moreover, a number of attacks have been proposed that expose the security vulnerabilities of these techniques. This paper highlights the key developments in the logic locking research and presents a comprehensive literature review of logic locking.**

I. Introduction

A. *IP piracy and reverse engineering*

The increasing cost of IC manufacturing has spawned an era of fabless semiconductor companies. With the outsourcing of IC fabrication in a globalized/distributed design flow including multiple (potentially untrusted) entities, the semiconductor industry is facing a number of challenging security threats such as IP piracy, overbuilding, reverse engineering, and hardware Trojans [1]–[4].

To address these security challenges [5], a number of hardware design-for-trust (DfTr) techniques such as IC metering [6], [7], watermarking [8], IC camouflaging [9]–[12], split manufacturing [13], [14], and logic locking [15]–[19] have been proposed. Logic locking, in particular, has received significant interest from the research community, as it is more versatile and can protect against a potential attacker located anywhere in the IC supply chain. The other DfTr techniques such as camouflaging or split manufacturing can protect only against a limited set of malicious entities, as illustrated in Table I.

B. *Logic locking*

Logic locking inserts additional logic into a circuit, locking the original design with a secret key. In addition to the original inputs, a locked circuit has *key inputs* that are driven by an on-chip tamper-proof memory, as illustrated in Figure 1. The additional logic may consist of XOR *key* gates [15]–[17] or look-up tables (LUTs) [20]. Figure. 2 presents the IC design flow incorporating logic locking. The locked netlist passes through the untrusted design phases. Without the secret key (i) the design details cannot be recovered (for reverse-engineering), and (ii) the IC is not functional, i.e., it produces incorrect outputs (for over-production). A locked IC needs to be activated by loading the secret key onto the chip's memory as illustrated in Figure 2.

Fig. 1: An abstract representation of a logic-locked design. Only on applying the secret key, the design produces correct outputs; otherwise, incorrect outputs are produced.

TABLE I: Protection offered by DfTr techniques against untrusted entities in the IC supply chain.

Technique	Foundry	SoC Integrator	Test	User
IC metering [6], [7]	✗	✓	✓	✓
Watermarking [8]	✗	✗	✗	✓
IC camouflaging [9]–[12]	✗	✓	✓	✓
Split manufacturing [13], [14]	✓	✗	✗	✗
Logic locking [15]–[19], [22], [25], [26]	✓	✓	✓	✓

C. *Evolution of logic locking: An overview*

Earlier logic locking efforts emphasized on the development of algorithms for deciding the best locations for inserting key gates, such as random [15], fault analysis-based [17], [20], and strong-interference-based logic locking [16]. Over the years, many key-recovery attacks have been mounted that exploit the vulnerabilities of logic locking techniques [16], [21], [22]. A summary of these attacks is presented in Table IV. A powerful attack that broke all the logic locking techniques existing then is *Boolean satisfiability (SAT)-based key-pruning attack*, referred to as *SAT attack*. The latest research works on logic locking have focused on defending against the SAT attack [18], [19], [23], [24]. While techniques such as Anti-SAT [18] and SARLock [19] thwart the SAT attack, they may be vulnerable to the emerging class of removal attacks [23].

D. *Contributions and organization of the paper*

This paper presents a review of the major developments in logic locking. Section II describes the terminology for logic locking and presents a classification of logic locking techniques. Section III presents a summary of the attacks that have been launched on logic locking. Section IV describes the traditional logic locking approaches. Section V elucidates the SAT attack resilient logic locking techniques. Section VI sheds light on emerging research directions.

II. Background

Before delving into further details of logic locking, we present a few formal definitions for the terms associated with logic locking followed by an example that illustrates the logic locking on an example netlist.

Original and locked netlists. The original netlist F is a Boolean function $F : I \rightarrow O$, where $I = \{0,1\}^n$ and $O = \{0,1\}^m$. The locked netlist is a Boolean function $L : I \times K \rightarrow O$, where $K = \{0,1\}^q$. Upon activation using the secret key k_s, $L(i, k_s) = F(i), \forall i \in I$.

Security of logic locking. A logic locking technique is considered secure if the computational effort required by a probabilistic polynomial time (PPT) adversary to determine the correct key value k_s, or equivalently, retrieve the original circuit functionality, is exponential in the size of the key: $\mathcal{O}(2^q)$.

Example of logic locking. Figure 3(a) presents an example (majority circuit) netlist, and Figure 3(b) shows its locked version with three XOR/XNOR key gates. One input of each

978-1-5386-2880-5/17/$31.00 ©2015 IEEE

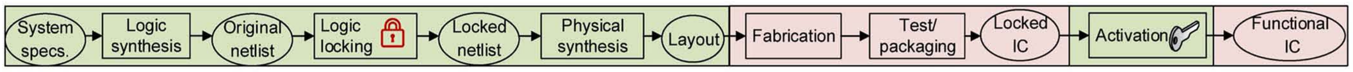

Fig. 2: Locking and activation of an IC in IC design flow. The red regions denote untrusted entities; the green regions represent trusted entities.

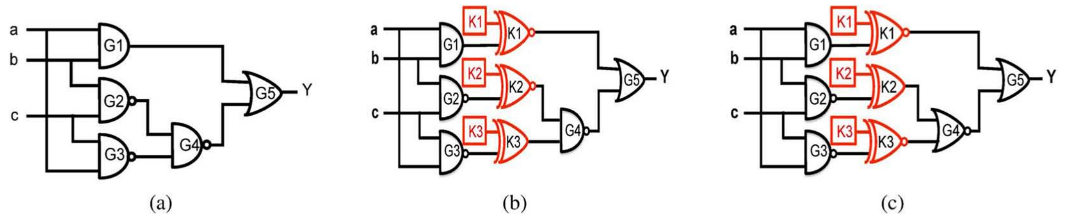

Fig. 3: Logic locking using XOR/XNOR gates [15]. a) An example circuit: majority of three inputs. b) Circuit locked using XOR/XNOR key gates. The correct key is value is 110. c) Locked circuit with inverters absorbed by the key gates. The correct key value is still 110.

key gate is driven by a wire in the original design, while the other input, referred to as key input, is driven by a key bit stored in a tamper-proof memory. To increase the obfuscation complexity, inverters can be added to or removed from the netlist; alternatively, the inverters in the netlist can be bubble-shifted to decouple they key value (i.e., 0/1) from the key gate type (i.e., XOR/XNOR) as illustrated in Figure 3(c). The locked IC (or a locked netlist) will not generate correct output unless it is activated using the correct key. Consider the locked circuit in Figure 3(b). When the correct key 110 is applied, the correct output $Y = 0$ is obtained for the input pattern 000. For the same input pattern, however, an incorrect output $Y = 1$ is produced for any incorrect key value.

A. Threat model

It is essential for any security scheme to specify the assumptions about the trusted/untrusted entities and the assets each entity has access to. As per logic locking threat model:

- The designer is trusted, i.e., the personnel and the tools used in the design house are trustworthy.
- The foundry, the test-facility, and the end-user are untrusted.
- The attacker has access to the locked netlist as well as a functional IC. The attacker also has the knowledge of the logic locking algorithm employed, as well as the location of the key gates. The only unknown is the secret key value, which is a binary vector.

B. Classification of logic locking

Logic locking can be classified into two major categories: *traditional* logic locking and *SAT attack resilient* logic locking.

1) **Traditional logic locking**. The techniques in this category focused on developing efficient algorithms for selecting the key gate locations. The traditional logic locking techniques include random [15], fault analysis based [17], and strong logic locking [27]. While most of the techniques employ *combinational* elements, there are also sequential logic locking techniques such as [20], [28], [29]. See Section IV for details.
2) **SAT attack resilient logic locking**. The research on logic locking took a whole new turn upon the inception of the SAT attack [22]; the attack was able to break all traditional logic locking techniques. Recent research efforts, such as SARLock [18], Anti-SAT [19], TTLock [30], and SFLL [24], focus on thwarting the SAT attack. See Section V for details.

C. Attacks on logic locking

Over the years, a number of attacks have been developed against logic locking techniques. We divide these attacks into following four categories: algorithmic attacks, approximate attacks, structural/removal attacks, and side-channel attacks.

1) **Algorithmic attacks**. These attacks exploit the algorithmic weaknesses of a logic locking algorithm to extract the secret unlock key. Since the secret key renders the functionality of the locked netlist "exactly" equivalent to the that of the functional IC, these attacks may also be referred to as "exact" attacks. Examples are the sensitization attack [16], the SAT attack [22], and the circuit partitioning attack [31].
2) **Approximate attacks**. Contrary to the exact attacks, the approximate attacks extract a netlist that is approximately the same as the original netlist, i.e., the netlist may produce incorrect output for only a few input patterns. This category of attacks includes AppSAT [32] and Double-DIP [33]. These attacks require lesser computational effort compared to the exact attacks.
3) **Structural attacks**. The fundamental principle of the structural/removal attacks is to bypass and/or remove the protection logic and isolate the functionally correct netlist. Examples include the signal probability skew attack attack [23], the AppSAT guided removal attack [26], and the Bypass attack [25].
4) **Side-channel attacks**. Side-channel attacks exploit the covert physical channels, such as power and timing, to leak the secret information [34]. It has been demonstrated that certain logic locking techniques may be susceptible to differential power analysis attack [35]. Test data is another side-channel that has been used to compromise the security of logic locking [21], [36]. A recent attack that can be categorized as a side-channel attack is the desynthesis attack [37]; the attack exploits the security vulnerabilities associated with logic synthesis.

III. Attacks on logic locking

In this section, we describe the various attacks on logic locking. Table II presents a summary of the attack threat models and methods.

A. Algorithmic attacks

Sensitization attack. Sensitization attack makes use of the VLSI test principle of sensitization [16]. Sensitization of a net w to an output o implies that the value of w can be propagated to o, in either true or complementary form. The attacker analyzes the locked netlist and computes attack patterns that can sensitize individual key bit values to primary

TABLE II: A summary of the attacks against logic locking.

Attack	Attacker assets	Attack method	Defense
Sensitization [16]	1) Locked netlist 2) Functional IC	Sensitization of key bits to circuit outputs	SLL [27]
SAT [22]	1) Locked netlist 2) Functional IC	SAT-based algorithm that rules out incorrect keys iteratively	Anti-SAT [19], SARLock [18], and SFLL [24]
AppSAT [32]	1) Locked netlist 2) Functional IC	Reduce a multi-layered defense to single-layered defense by augmenting SAT attack with random oracle queries	Anti-SAT [19], SARLock [18], SFLL [24]
Double-DIP [33]	1) Locked netlist 2) Functional IC	Reduce a multi-layered defense to single-layered defense by using DIPs that eliminate at least two incorrect keys	Anti-SAT [19], SARLock [18], SFLL [24]
Signal probability skew (SPS) [23]	1) Locked netlist	Trace the Anti-SAT block using signal skew and remove it	SFLL [24]
AppSAT guided removal (AGR) [26]	1) Locked netlist 2) Functional IC	Use AppSAT to find FLL key bits; trace keys to identify and remove Anti-SAT	SFLL [24]
Bypass [25]	1) Locked netlist 2) Functional IC	Find all the DIPs for a random key and correct the output accordingly	SFLL [24], obfuscated Anti-SAT [19]
Hill climbing [21]	1) Locked netlist 2) Test data	Start with a random key CK. Flip the bits in CK based on the Hamming distance	Test-aware logic locking [21]
Test-data mining [36]	1) Locked netlist 2) Test data	Find the key that maximizes fault coverage and satisfies test data constraints	Post-test activation [36]
Differential power analysis [35]	1) Locked netlist 2) Functional IC	Generate a differential trace from power samples for each key value	Anti-SAT [19], SARLock [18], and SFLL [24]
Desynthesis [37]	1) Locked netlist	Generate a resynthesized netlist with maximum similarity to the locked netlist	Meerkat [37]

outputs. These attack patterns are then applied to the functional IC; by analyzing the observed IC responses, an attacker can determine the value of key bits.

SAT attack. The SAT attack is based on the notion of specialized input patterns known as distinguishing input patterns (DIPs) [22]. The attack rules out incorrect key values by using DIPs iteratively. A DIP is an input value for which two unique key values, k_1 and k_2, produce differing outputs, o_1 and o_2, respectively. Since o_1 and o_2 are different, at least one of the key values is incorrect; the functional IC output is used to identify which one(s). A single DIP may rule out multiple incorrect key values, reducing the computational effort of the attack.

Example. Table III represents the output values of the locked circuit with three primary and three key inputs for different key/input combinations. The values $(k0, \ldots, k7)$ represent all possible values for three key inputs $\{K1, K2, K3\}$. When the attack is launched, it takes four DIPs to obtain the correct key. The last column in the table lists the keys eliminated in each iteration. For example, in iteration 4, the pattern 010 is used to eliminate all incorrect keys, thus identifying $k5$ as the correct key.

Circuit partitioning (CP) attack. The CP attack operates in a divide-and-conquer fashion [31]. The attack divides a circuit into logic cones and targets individual logic cones using brute-force. This divide-and-conquer approach is also utilized by the DPA attack [35].

TABLE III: Analysis of the SAT attack [22] against random logic locking [15]. The red entries represent the keys identified as incorrect. k5 is the correct key; the columns with all correct output values are shaded blue.

abc	Y	k0	k1	k2	k3	k4	k5	k6	k7	Incorrect keys identified
000	0	1	1	1	1	1	0	1	1	
001	0	1	1	1	1	1	0	1	1	
010	0	1	1	1	1	1	0	1	1	iter 4: other incorrect keys
011	1	1	1	1	1	1	1	1	1	
100	0	1	1	1	1	1	0	1	0	
101	1	1	0	1	1	1	1	1	1	iter 3: k1
110	1	1	1	1	1	0	1	1	1	iter 1: k4
111	1	1	1	0	1	1	1	1	1	iter 2: k2

B. Approximate attacks

AppSAT. AppSAT targets a compound logic locking technique, e.g., SARLock+SLL or Anti-SAT+SLL, and reduces it to a single SAT attack resilient technique [32]. As such, the attack is not applicable to SARLock or Anti-SAT alone; the attack will terminate unsuccessfully within a few iterations. The attack works by augmenting the SAT attack with random queries to the functional IC at regular intervals. The attack cautions about the naive integration of multiple logic locking techniques; the overall design may still be insecure.

Double-DIP. The Double-DIP attack also targets compound logic locking techniques [33]. The attack computes DIPs that eliminate at least two incorrect keys in each iteration. When the attack fails to find such DIPs, the compound technique (e.g., SARLock+SLL) has been reduced to a single SAT attack resilient technique (SARLock in this case).

C. Structural attacks

Signal probability skew (SPS) attack . The basic (unobfuscated) Anti-SAT block (see Section V) comprises an AND-tree and a NAND-tree, whose outputs are skewed towards 0 and 1, respectively [19]. The SPS attack exploits these structural traces – the skew in the signal probabilities – to locate and isolate the protection logic [23]. The attack becomes less effective in the presence of structural/functional obfuscation.

AppSAT guided removal (AGR) attack. To isolate the Anti-SAT block that has been obfuscated with additional XOR and multiplexer key gates, the AGR attack integrates AppSAT with simple post-processing steps [26]. The AppSAT identifies most of the key bits meant for obfuscation; the remaining key bits belong to Anti-SAT block. The post-processing steps identify the gate at which most of the Anti-SAT key bits converge; moreover, the ratio of key bits converging at each of its inputs is almost 0.5. The gate constitutes the output Y of the Anti-SAT block. The attacker can then re-synthesize the design with a constant 0 (1) on the signal Y, retrieving the original design.

Bypass attack. Contrary to the SPS or AGR attack, the Bypass attack adds *bypass* circuitry around the locked netlist [25]. The attack assigns a random key to the locked

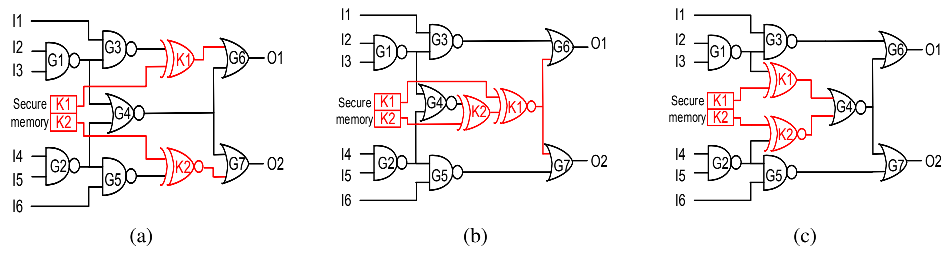

Fig. 4: Key gate insertion based on a) random [15], b) fault analysis based [17], and c) strong logic locking [27].

netlist and determines the DIP(s) for which the netlist outputs are incorrect; the bypass circuit is designed to restore the output for the computed DIPs.

D. Side-channel attacks

Test-data attacks. While the typical attacks on logic locking rely on input/output observations recorded from a functional IC, test-data attacks, such as hill climbing attack [21] and test-data mining attack [36], extract secret information from the test data. Test data is generated at the design house using automatic test pattern generation (ATPG) tools, sent to the foundry/dedicated test facility, and utilized during manufacturing test to classify ICs as faulty/fault-free. The same ATPG tools can be used by an attacker in the test facility to leak the secret logic locking key.

Differential power analysis (DPA) attack. The DPA attack is one of the most powerful side-channel attacks that have been used to break most of the cryptographic algorithms [34]. DPA attack on logic locking is launched by collecting power samples and the ciphertext output from the IC under attack for a large set of plaintext inputs [35]. The collected samples are then analyzed using statistical analysis yielding a differential trace, which tends to be high for the correct key value and zero for the incorrect key values [35]. The DPA attack may be launched on the individual logic cones in a divide-and-conquer approach.

Desynthesis attack. The desynthesis attack relies on the observation that the locked and the original netlist should be similar in terms of the type and count of the gates [37]. The attack re-synthesizes the locked netlist with a random key and then uses hill climbing search to find the key value that yields the maximum similarity between the locked netlist and the re-synthesized netlist. The attack can be thwarted using Meerkat, a BDD based logic locking technique that inserts two nodes for each key bit in a desynthesis-agnostic approach [37].

IV. Traditional Logic Locking

In this section, we describe the traditional logic locking techniques.

Random logic locking (RLL). The random logic locking technique [15] locks a design by inserting XOR key-gates at random locations in a netlist. Figure 4(a) shows an example netlist locked with two key-gates, K1 and K2, using random logic locking. In random logic locking, the key gates are spread uniformly in the entire netlist. The interference among the key gates tends to be minimal, rendering random logic locking vulnerable to attacks.

Fault analysis based logic locking (FLL). FLL aims at preventing the black-box usage of an IC [17]. In random logic locking, even incorrect keys may lead to correct output for certain input patterns. FLL ensures maximum corruption at the output bits when incorrect keys are used. The output corruption is measured in terms of percentage Hamming distance between the correct output and the incorrect output, obtained upon applying incorrect keys.

The key gates are inserted at the most *influential* locations in the circuit, i.e., the locations that have the maximum impact on the output when incorrect key values are applied. Figure 4(b) shows a netlist locked using fault analysis based logic locking. The key gates tend to be inserted in back-to-back fashion, which undermines the security of logic locking as this creates multiple correct key values [16].

While Rajendran et al. [17] used VLSI test based algorithms to maximize output corruption, Colombier et. al show that for certain circuits, the same objectives can be achieved using graph centrality indicators with significantly lower computational effort [38].

Strong logic locking (SLL). SLL thwarts the sensitization attack by inserting key gates with maximum mutual interference and prevents sensitization of the key bits on an individual basis [16]. As the sensitization of key bits is hampered, an attacker is forced handle multiple key bits together rather than individual [27]. Consider the netlist in Figure 4(c). The netlist has two key-gates, K1 and K2, inserted using the SLL algorithm. It can be seen that K1 and K2 interfere each other's path to the primary outputs. It is not possible for an attacker to sensitize either K1 or K2 to a primary output individually.

V. SAT Attack Resilient Logic Locking

The techniques developed recently to mitigate the SAT attack include SARLock [18], Anti-SAT [19], TTLock [30], and SFLL [24]. Figure 5 illustrates the recent SAT attack resilient logic locking techniques. The underlying idea of most of these techniques is to utilize point functions [1] to control the amount of error injected into a circuit on the application of incorrect key values.

SARLock. As shown in Figure 5(a), SARLock protection circuitry comprises a comparator and a mask block that are integrated with the original circuit [18]. For the correct key value, no error is injected into the circuit, and the correct output is retained. For each incorrect key value, an error is injected into the circuit for only one input pattern, leading to an incorrect output for the specific pattern. Assuming that $F(I)$ is the original circuit, the output O of the circuit locked using SARLock can be presented as $O = F(I) \oplus ((I == K) \oplus (I == k_s))$, where K denotes the key inputs, and k_s is the correct key value.

Anti-SAT. The Anti-SAT block shown in Figure 5(b) is constructed using two complementary blocks, $B_1 = g(X, K_{l1})$ and $B_2 = \overline{g(X, K_{l2})}$ [19]. These blocks share the same inputs

[1]A point function is a Boolean function that produces the output value 1 at exactly one point. Example implementations include AND gates and password checkers.

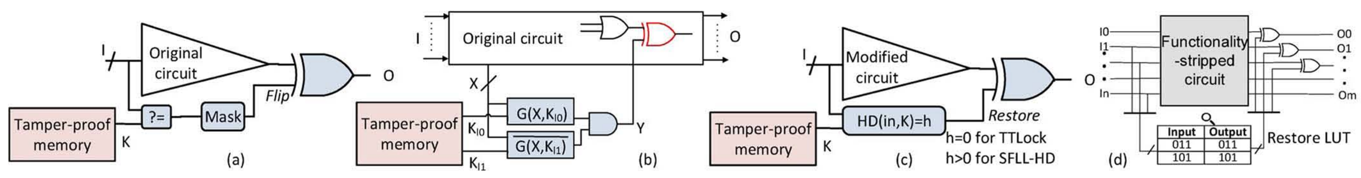

Fig. 5: SAT attack resilient logic locking techniques: a) SARLock [18], b) Anti-SAT [19], c) SFLL-HD [24]/TTLock [30], and d) SFLL-flex [24].

TABLE IV: Attack resiliency of logic locking techniques. ✗ denotes attack success, ✓ denotes resilience, and ✗ * denotes partial success.

Attack	RLL [15]	FLL [17], [20]	SLL [16]	AntiSAT [19]	SARLock [18]	SFLL [24]	Compound locking (e.g. Anti-SAT+SLL)
Sensitization [16]	✗	✗	✓	✓	✓	✓	✓
SAT [22]	✗	✗	✗	✓	✓	✓	✓
AppSAT [32]	✗	✗	✗	✓	✓	✓	✗ *
Double-DIP [33]	✗	✗	✗	✓	✓	✓	✗ *
SPS [23]	✓	✓	✓	✗	✗	✓	✓
AGR [26]	✓	✓	✓	✗	✗	✓	✗
Bypass [25]	✓	✓	✓	✗	✗	✓	✓
Test-data mining [36]	✗	✗	✗	✓	✓	✓	✓
Hill climbing [21]	✗	✗	✓	✓	✓	✓	✓
DPA [35]	✗	✗	✗	✓	✓	✓	✓
Desynthesis [37]	✗	✗	✗	✓	✓	✓	✓

X, but are locked with different keys K_{l1} and K_{l2}. The outputs of B_1 and B_2 drive an AND gate to produce the output signal Y. The two blocks produce complementary outputs when the correct key value is applied; for all inputs, $Y = 0$, leading to a correct output. For an incorrect key value, the output of B_1 and B_2 is 1 for a specific input pattern; for that pattern, $Y = 1$, leading to an incorrect output. Assuming that Anti-SAT protects one of the primary outputs of the original circuit $F(I)$, the protected output O can be represented as $O = F(I, K_{l0}) \oplus (g(X \oplus K_{l1}) \wedge \overline{g(X \oplus K_{l2})})$, where K_{l0} represents the key for the logic locked circuit.

Tenacious and tracelss logic locking (TTLock). Both SARLock and Anti-SAT are vulnerable to structural attacks since they implement the original function as is. In TTLock, the original logic cone $F(I)$ is modified for exactly one input pattern i_s to hide the true implementation from an attacker and thwart structural attacks [30]. The output of the logic cone for the *protected* input pattern is then restored using a protection circuit that is essentially a comparator block. TTLock is illustrated in Figure 5(c). Upon removal attack, the attacker retrieves a netlist which differs from the original netlist for exactly one input pattern.

Stripped functionality logic locking (SFLL). Stripped functionality logic locking resists all known attacks against logic locking, and enables trade-offs between resilience to SAT attack and the removal attacks [24]. SFLL is based on the notion of "strip and restore", where some functionality from the original circuit is stripped and is stored in form secret keys in an on-chip tamper-proof memory. SFLL has two variants: SFLL-HD and SFLL-flex.

SFLL-HD. Whereas TTLock modifies and thus protects only one input pattern, SFLL-HD allows to efficiently protect a large number of input patterns [24]. SFLL-HD is able to protect $\binom{k}{h}$ input patterns that are of Hamming Distance (HD) h from the k-bit secret key. Only one k-bit secret key is stored in the tamper-proof memory. As depicted in Figure 5(c), a single comparator is used along with the Hamming distance compute logic. With increasing Hamming distance, the number of protected patterns increased binomially. The SAT attack resilience decreases logarithmically with increasing number of protected patterns. For h=0, SFLL-HD is equivalent to TTLock.

SFLL-flex. SFLL-HD is suitable for general applications where it is useful to protect an arbitrary set of input patterns. However, in certain applications, a specified set of input patterns, or a range of input patterns needs to be protected. SFLL-flex allows to compactly represent the patterns-to-be-protected using a small set of input cubes [24][2]. The input cubes are stored on an on-chip look-up table as illustrated in Figure 5(d).

VI. Discussion

Attack/defense matrix. Table IV presents the resiliency of various logic locking countermeasures against each of the aforementioned attacks. The SAT attack [22] breaks all traditional logic locking defenses. Thus, any incremental improvement over these defenses, such as achieving better Hamming distance [38] or improved key-interference at lower overhead [39], are still susceptible to the attack. Among the SAT resilient logic locking techniques, only SFLL resists all known attacks on logic locking. Every other technique is vulnerable to at least one attack.

Emerging research directions. This paper emphasizes a strong need to develop holistic logic locking solutions with provable security against all known and anticipated attacks. The security of the SAT resilient logic locking techniques against the synthesis attacks needs further investigation, since techniques such as SFLL rely on existing synthesis tools.

Another area of research that warrants more investigation is side-channel attacks. DPA attack is only one example of side-channel attacks; the other side channel attacks such as timing need to be studied in the context of logic locking. This holds particularly true for the emerging delay locking [40] technique, where the incorrect key values affect the timing of the design. Parametric locking is another emerging paradigm for logic locking; apart from functionality, it is possible to lock performance and power, etc. [30], [41].

VII. Conclusion

The paper summarizes the key developments in the field of logic locking. While the initial focus of the logic locking

[2]Input cubes refer to partially-specified input patterns; some input bits are set to logic-0's or logic-1's, while other input bits are don't cares (x's).

research was on efficient gate selection strategies, the focus has gradually shifted towards simultaneously thwarting the SAT attack as well as structural attacks. It is essential to evaluate a logic locking technique against all known as well as anticipated attacks and offer provable security guarantees. An emerging paradigm for logic locking is parametric locking where performance, power etc. may be locked.

VIII. Acknowledgement

This work was supported by the New York University/New York University Abu Dhabi (NYU/ NYUAD) Center for Cyber Security (CCS).

References

[1] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[2] R. Torrance and D. James, "The State-of-the-Art in Semiconductor Reverse Engineering," in *IEEE/ACM Design Automation Conference*, 2011, pp. 333–338.

[3] Chipworks, "Intel's 22-nm Tri-gate Transistors Exposed," http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed/, 2012.

[4] M. M. Tehranipoor, U. Guin, and S. Bhunia, "Invasion of the Hardware Snatchers," *IEEE Spectrum*, vol. 54, no. 5, pp. 36–41, 2017.

[5] T. S. Perry, "Why Hardware Engineers Have to Think Like Cybercriminals, and Why Engineers Are Easy to Fool," http://spectrum.ieee.org/view-from-the-valley/computing/embedded-systems/why-hardware-engineers-have-to-think-like-cybercriminals-and-why-engineers-are-easy-to-fool, 2017.

[6] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security," in *USENIX Security*, 2007, pp. 291–306.

[7] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 51–63, 2012.

[8] A. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection," in *IEEE/ACM Design Automation Conference*, 1998, pp. 776–781.

[9] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security Analysis of Integrated Circuit Camouflaging," in *ACM/SIGSAC Conference on Computer & Communications Security*, 2013, pp. 709–720.

[10] A. Vijayakumar, V. Patil, D. 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*, vol. 12, no. 1, pp. 64–77, 2017.

[11] M. Massad, S. Garg, and M. Tripunitara, "Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes," in *Network and Distributed System Security Symposium*, 2015.

[12] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "CamoPerturb: Secure IC Camouflaging for Minterm Protection," in *IEEE/ACM International Conference on Computer-Aided Design*, 2016, pp. 29:1–29:8.

[13] R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semiconductor Circuits," 2007, US Patent 7,195,931.

[14] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in *USENIX Conference on Security*, 2013, pp. 495–510.

[15] J. Roy, F. Koushanfar, and I. L. Markov, "Ending Piracy of Integrated Circuits," *IEEE Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[16] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *IEEE/ACM Design Automation Conference*, 2012, pp. 83–89.

[17] J. Rajendran, H. Zhang, C. Zhang, G. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault Analysis-Based Logic Encryption," *IEEE Transactions on Computer*, vol. 64, no. 2, pp. 410–424, 2015.

[18] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT Attack Resistant Logic Locking," in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2016, pp. 236–241.

[19] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2016, pp. 127–146.

[20] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Des. Test. Comput.*, vol. 27, no. 1, pp. 66–75, 2010.

[21] S. Plaza and I. Markov, "Solving the Third-Shift Problem in IC Piracy With Test-Aware Logic Locking," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[22] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2015, pp. 137–143.

[23] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security Analysis of Anti-SAT," *IEEE Asia and South Pacific Design Automation Conference*, pp. 342–347, 2016.

[24] M. Yasin, A. Sengupta, M. Ashraf, M. Nabeel, J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *ACM/SIGSAC Conference on Computer & Communications Security*, 2017, pp. 1–1, to appear.

[25] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against all Known Logic Locking Attacks," Cryptology ePrint Archive, Report 2017/621, 2017, http://eprint.iacr.org/2017/621.

[26] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 99, no. 0, p. PP, 2017, to appear.

[27] M. Yasin, 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*, vol. 35, no. 9, pp. 1411–1424, 2016.

[28] R. S. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Compute-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, 2009.

[29] S. Khaleghi, K. D. Zhao, and W. Rao, "IC Piracy Prevention via Design Withholding and Entanglement," in *Asia Pacific Design Automation Conference*, 2015, pp. 821–826.

[30] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to lock?: Functional and parametric locking," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, ser. GLSVLSI '17, 2017, pp. 351–356.

[31] Y.-W. Lee and N. Touba, "Improving Logic Obfuscation via Logic Cone Analysis," in *Proc. Latin-American Test Symposium*, 2015, pp. 1–6.

[32] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z., and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *to appear in IEEE International Symposium on Hardware Oriented Security and Trust*, 2017.

[33] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," Cryptology ePrint Archive, Report 2017/290, 2017, http://eprint.iacr.org/2017/290.

[34] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology*. Springer, 1999, pp. 388–397.

[35] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security Analysis of Logic Encryption against the Most Effective Side-Channel Attack: DPA," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2015, pp. 97–102.

[36] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of Logic Encrypted Chips: Pre-test or Post-Test?" in *Design, Automation Test in Europe*, 2016, pp. 139–144.

[37] M. E. Massad, J. Zhang, S. Garg, and M. V. Tripunitara, "Logic Locking for Secure Outsourced Chip Fabrication: A New Attack and Provably Secure Defense Mechanism," *CoRR*, vol. abs/1703.10187, 2017. [Online]. Available: http://arxiv.org/abs/1703.10187

[38] B. Colombier, L. Bossuet, and D. Hely, "Centrality Indicators for Efficient and Scalable Logic Masking," in *IEEE Computer Society Annual Symposium on VLSI*, 2017, pp. 98–103.

[39] R. Karmakar, N. Prasad, S. Chattopadhyay, R. Kapur, and I. Sengupta, "A New Logic Encryption Strategy Ensuring Key Interdependency," in *International Conference on VLSI Design*, 2017, pp. 429–434.

[40] Y. Xie and A. Srivastava, "Delay Locking: Security Enhancement of Logic Locking against IC Counterfeiting and Overproduction," in *IEEE/ACM Design Automation Conference*, 2017, pp. 9:1–9:6.

[41] L. Li and H. Zhou, "Structural transformation for best-possible obfuscation of sequential circuits," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2013, pp. 55–60.