Chapter 2
A Brief History of Logic Locking

Abstract This chapter presents a comprehensive history of logic locking defenses and attacks. A classification of logic locking techniques as well as attacks is provided. The logic locking defenses are divided into classes: pre-SAT and post-SAT techniques. Four classes of attacks: algorithmic, approximate, structural, and side-channel, are introduced. The chapter emphasizes the relationship between different logic locking techniques. A timeline of the prominent logic locking attacks and defenses is also presented.

Since the inception of logic locking in 2008 [16], it has received significant interest from the research community. Over the last decade, a number of logic locking techniques as well as attacks have emerged. This chapter presents a high-level introduction to the major developments in logic locking. In addition to offering an overview of different logic locking attacks and defenses, the chapter also highlights the relationships between different attack and defense algorithms. Section 2.1 provides a summary of the milestones in logic locking research. Section 2.2 introduces a classification of logic locking defenses and attacks. Section 2.3 presents a brief overview of the existing logic locking defenses. Section 2.4 summarizes the existing attacks on logic locking. Section 2.5 elaborates on the resilience of each defense against different attack algorithms using an attack-defense matrix. Section 2.6 presents a summary of different metrics that can be used to evaluate the effectiveness of a logic locking technique.

2.1 Milestones in Logic Locking

This section focuses on the milestones in logic locking research. We introduce the first logic locking defense, describe the first attack against it, and also discuss the most powerful attack on logic locking. While detailed descriptions of these attacks and defense techniques are presented in the relevant chapters, only an

M. Yasin et al., *Trustworthy Hardware Design: Combinational Logic Locking Techniques*, Analog Circuits and Signal Processing,
https://doi.org/10.1007/978-3-030-15334-2_2

abstract overview is presented here so that the reader becomes familiar with the key advancements in the field of logic locking.

2.1.1 The First Defense

The first logic locking technique, introduced in 2008, is random logic locking (RLL) [16]. As the name indicates RLL inserts XOR/XNOR key gates at random locations in a netlist; only upon supplying the correct key to a locked chip, it becomes functional. To put things in a historical perspective, it must be mentioned that RLL was introduced in the context of IC metering. The overall IC metering framework EPIC, which abbreviates "Ending Piracy of Integrated Circuits", not only locks the design with a common locking key CK but also provides additional circuitry for generating unique key values for each manufactured IC. EPIC also makes use of public-key cryptography and enables remote activation of a chip. A designer can communicate remotely with the chip fabricated at an untrusted foundry, and can securely load the key CK onto the chip. While an interested reader can refer to [16, 17] for details of the EPIC protocol, it suffices to say that at the core of EPIC is RLL that renders the functionality of the fabricated chip dependent on the key inputs. We discuss RLL in detail in Sect. 3.1.

2.1.2 The First Threat Model and Attack

While RLL was the first defense logic locking defense and inspired further research on IC/IP piracy [1, 13], it was discovered by Rajendran et al. [14] in 2012 that the individual key bits in the common key CK that activates a functional IC may be observed on the primary outputs of IC. The new threat model introduced by Rajendran et al. assumes that the attacker has access to two critical assets, (1) a reverse-engineered netlist, and (2) a functional IC. This threat model has been adopted by all subsequent logic locking attacks and defenses. Their proposed *sensitization* attack, which follows the aforementioned threat model proceeds as follows. By analyzing the locked netlist, the attack computes attack patterns. By applying these patterns to a functional IC, specific key bits may be sensitized to primary outputs of the IC, thus leaking the secret key. In RLL, most key bits do not interfere in each other's path to the primary outputs, and can be targeted on an individual basis [13].

As a countermeasure against the sensitization attack, strong logic locking (SLL) was introduced that inserts key-gates in a way that they protect one another [14] (see Sect. 3.4.3 for details).

2.1.3 The Most Powerful Attack

Apart from RLL and SLL, other important research efforts on logic locking include fault-analysis based logic locking (FLL) [13] and look-up table based locking [1]. However, in 2015, Pramod et al. [21] developed a powerful attack on logic locking that could break all the defense techniques existing then. The attack utilizes a Boolean satisfiability (SAT) formulation to encode the problem of finding the logic locking key and is commonly referred to as the *SAT attack*. The attack uses specific *distinguishing input patterns (DIPs)* to refine the key search space iteratively. The SAT attack has drastically changed the focus of logic locking research; most recent research efforts aim at developing effective countermeasures against the SAT attack [10, 22, 25, 30]. Chapter 4 discusses the SAT attack in further details.

2.2 Classification of Attacks and Defenses

2.2.1 Classifying Logic Locking Techniques

There can be multiple ways to classify the existing logic locking techniques. For instance, based on the type of logic elements constituting the protection circuitry, logic locking techniques can broadly be classified as either combinational or sequential. Whereas the combinational logic locking techniques insert XOR/XNOR gates [14, 15, 17], AND/OR gates [4], or multiplexers [12, 15] into a circuit, the sequential logic locking techniques insert look-up tables (LUTs) [1, 6], or finite state machines (FSMs) [2, 8]. Most of the recent research efforts focus on combinational logic locking techniques, as the tools and methods employed for the security analysis of the combinational techniques are mature and easily available. More importantly, the sequential circuits can often be treated as combinational by leveraging the test infrastructure on a chip, with scan chains being the most commonly used test access mechanism [12, 17]. This book focuses on combinational logic locking.

We classify the combinational logic locking techniques into major classes, (1) Pre-SAT and (2) Post-SAT logic locking. This classification is relative to the inception of the SAT attack [21]. Pre-SAT logic locking techniques are those that have been developed prior to the SAT attack and remain susceptible to the attack. Post-SAT techniques are those that have been developed after the SAT attack with the objective of thwarting the attack.

1. **Pre-SAT logic locking**. The techniques in the pre-SAT era focused on developing algorithms for selecting the key gate locations. These basic logic locking techniques include random logic locking (RLL) [17], fault analysis based logic locking (FLL) [15], and strong logic locking (SLL) [27].
2. **Post-SAT logic locking**. The research on logic locking took a whole new turn upon the inception of the SAT attack [21]; the attack was able to break all tradi-

tional logic locking techniques. Recent research efforts, such as SARLock [25], Anti-SAT [22], TTLock [31], and SFLL [30], focus on thwarting the SAT attack.

An introductory overview of various logic locking techniques is presented in Sect. 2.3; further details about each technique are furnished in the relevant chapters.

2.2.2 Classifying Attacks on Logic Locking

Over the years, a number of attacks have been developed against logic locking techniques. We divide these attacks into the 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 key. Since the secret key renders the functionality of the locked netlist "exactly" equivalent to that of the functional IC, these attacks may also be referred to as "exact" attacks. Examples are the sensitization attack [14], the SAT attack [21], and the circuit partitioning (CP) attack [9].
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 an incorrect output for only a few input patterns. This category of attacks includes AppSAT [19] and Double-DIP [20]. 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 [26], the AppSAT guided removal attack [29], and the Bypass attack [23].
4. **Side-channel attacks**. Side-channel attacks exploit the covert physical channels, such as power and timing, to leak the secret information [7]. It has been demonstrated that certain logic locking techniques may be susceptible to differential power analysis attack [24]. Test data is another side-channel that has been used to compromise the security of logic locking [12, 28]. A recent attack that can be categorized as a side-channel attack is the desynthesis attack [11]; the attack exploits the security vulnerabilities associated with logic synthesis.

A brief introduction to the aforementioned attacks is provided in Sect. 2.4; further details are presented in the subsequent chapters.

2.2.3 A Timeline of Logic Locking

Figure 2.1 presents a timeline view of prominent logic locking attacks and defenses. The defenses are presented above the timeline and the attacks below it. The color of each dot represents the class of an attack or a defense. Following the classification

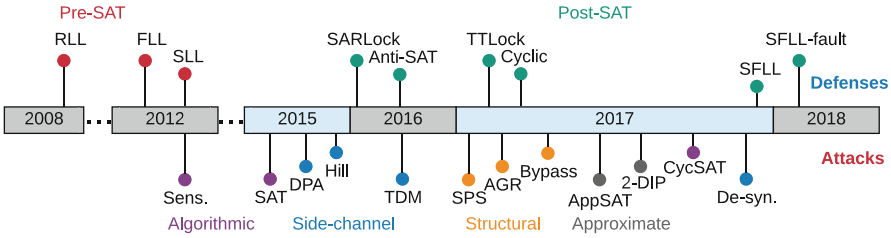

Fig. 2.1 A timeline view of logic locking attacks and defenses

mentioned above, there are two classes of defenses and four classes of attacks. It can be observed that the logic locking research has taken off 2015 on-wards with the emergence of multiple attacks and defenses. We believe that presenting this timeline earlier in the chapter puts things in a better perspective for the reader. Towards the end of this chapter, we also present an attack-defense matrix that elaborates on the connection between different attacks and countermeasures.

2.3 An Overview of Existing Defenses

2.3.1 Pre-SAT Logic Locking

All pre-SAT logic locking techniques insert either XOR/XNOR key gates or MUX key gates at selected locations in a netlist. These techniques differ mainly in the key gate location selection algorithm. As already mentioned, there exist three basic logic locking techniques: RLL [17], FLL [1, 15], SLL [14, 27]. Other gate selection algorithms such as [3–5] can be considered as variants of the three basic techniques. Here, we only present a brief description of the basic logic locking techniques. The detailed algorithms and illustrative examples are presented in Chap. 3.

RLL As already pointed out in the discussion of logic locking milestones, RLL inserts XOR/XNOR key gates at random locations in a netlist [17]. Figure 2.2a presents a netlist locked using RLL. RLL remains vulnerable to sensitization and SAT attacks [14, 21].

FLL A limitation of RLL is that even incorrect key values may lead to a correct circuit output for a large fraction of input patterns, enabling the black-box usage of a chip. FLL aims at preventing black-box usage of a locked chip [15]. It ensures that the maximum error is observed at the circuit output upon the application of incorrect key values. The *output corruption* is measured in terms of the percentage Hamming distance between the correct output and the incorrect output, obtained upon applying incorrect keys.

In FLL, the key gates are inserted at the most *influential* locations in the circuit, i.e., the locations that exhibit the highest impact on the circuit output upon the

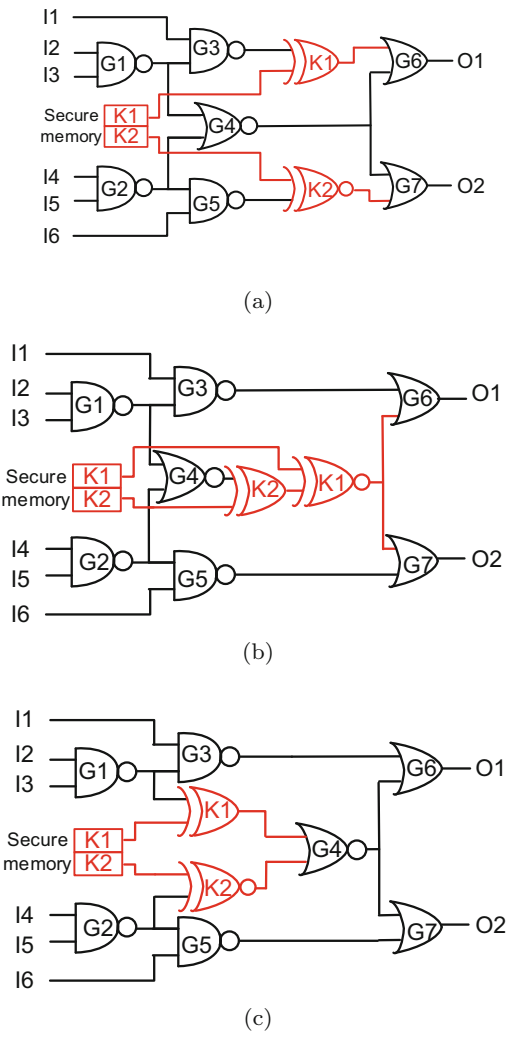

Fig. 2.2 Key gate insertion based on basic logic locking techniques. (**a**) Random [17], (**b**) fault analysis-based [15], and (**c**) strong logic locking [27]

application of incorrect key values. Figure 2.2b shows a netlist locked using FLL. FLL is described in more detail in Sect. 3.2.

SLL As pointed earlier, SLL aims at thwarting the sensitization attack by inserting key gates in a way that maximizes the interference among the key-gates and prevents the sensitization of the key bits on an individual basis. With increased interference among the key-gates, an attacker is forced to brute-force an exponentially growing number of key combinations [14].

Consider the netlist in Fig. 2.2c with two key-gates, K1 and K2, that are inserted using SLL. It can be seen that K1 and K2 interfere with 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 on an individual basis. Refer to Sect. 3.4.3 for further details on SLL.

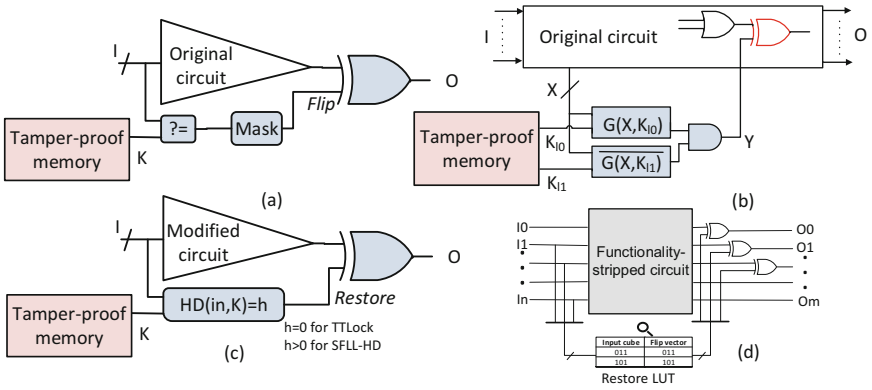

Fig. 2.3 SAT attack resilient logic locking techniques: (**a**) SARLock [25], (**b**) Anti-SAT [22], (**c**) SFLL-HD [30]/TTLock [31], and (**d**) SFLL-flex [30]

2.3.2 Post-SAT Logic Locking

The techniques developed recently to mitigate the SAT attack include SARLock [25], Anti-SAT [22], ATD [10], TTLock [31], and SFLL [30]. Figure 2.3 illustrates the recent SAT attack resilient logic locking techniques. The common denominator for most of these techniques is a point function, which injects error for selectively into a circuit in a way that thwarts the SAT attack. 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. Point functions help control the amount of error injected into a circuit upon the application of incorrect key values.

SARLock As shown in Fig. 2.3a, SARLock protection circuitry comprises a comparator and a mask block that are integrated with the original circuit [25]. 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 (refer to Sect. 5.2 for details).

Anti-SAT The Anti-SAT block shown in Fig. 2.3b is constructed using two complementary blocks, $B_1 = g(X, K_{l1})$ and $B_2 = \overline{g(X, K_{l2})}$ [22]. These blocks share the same inputs 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 (refer to Sect. 5.3 for details).

AND-Tree Detection (ATD) As opposed to Anti-SAT that integrates external point functions with the original netlist, ATD identifies such structures inside an original netlist and reuses them to decrease the implementation cost [10]. Once an AND (OR)-tree has been identified in a netlist, it is locked by inserting XOR/XNOR key gates at its inputs (refer to Sect. 5.4 for details).

Tenacious and Traceless Logic Locking (TTLock) Both SARLock and Anti-SAT are vulnerable to structural attacks since they implement the original function as is [31]. 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 [31]. 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 Fig. 2.3c. Upon removal attack, the attacker retrieves a netlist which differs from the original netlist for exactly one input pattern. A limitation of TTLock is that it protects only one input pattern, resulting in (1) the minimal removal attack resilience, and (2) the minimal error injection. For any input pattern, only two key values inject an error into the circuit. (refer to Sect. 9.2 for details).

SFLL-HD Whereas TTLock modifies and thus protects only one input pattern, SFLL-HD allows to efficiently protect a large number of input patterns [30], leading to a higher removal attack resilience and a higher error injection rate. SFLL-HD is able to protect $\binom{k}{h}$ input patterns that are of Hamming distance h from the k-bit secret key. Only one k-bit secret key is stored in the tamper-proof memory. As depicted in Fig. 2.3c, a single comparator is used along with the Hamming distance checker. With increasing h, the number of protected patterns increases binomially. The SAT attack resilience decreases logarithmically with increasing number of protected patterns. For $h = 0$, SFLL-HD is equivalent to TTLock (refer to Sect. 9.3 for details).

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 [30].[1] The input cubes are stored on an on-chip look-up table as illustrated

[1] 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).

in Fig. 2.3d. Only upon loading the correct cubes to the LUT, the circuit becomes functional (refer to Sect. 9.4 for details).

While point function-based techniques thwart the SAT attack by increasing the number of iterations required for the attack to succeed, ORF-Lock makes use of one-way functions to increase the time for individual attack iterations (refer to Sect. 8.3 for details). CycSAT is another SAT attack resilient technique that introduces cycles into a locked circuit, rendering the traditional SAT attack ineffective (refer to Sect. 8.1 for details).

2.4 Logic Locking Attacks

2.4.1 Algorithmic Attacks

Having already introduced the sensitization attack and the SAT attack in Sect. 2.1, we introduce only the CP attack and the CycSAT attack here.

Circuit Partitioning (CP) Attack The CP attack operates in a divide-and-conquer fashion [9]. The attack divides a circuit into logic cones and targets individual logic cones using brute-force. A logic cone is a sub-circuit consisting of gates that are in the transitive fan-in of a specific primary output. Depth-first search may be used to construct a logic cone. This divide-and-conquer approach is also utilized by the DPA attack [24].

CycSAT The CycSAT attack circumvents cyclic logic locking. The attack builds on top of the SAT attack; it adds additional constraints to the SAT formula that help eliminate the cycles from the netlist [32].

2.4.2 Structural Attacks

Signal Probability Skew (SPS) Attack The basic Anti-SAT block comprises an AND-tree and a NAND-tree, whose outputs are skewed towards 0 and 1, respectively [22]. The SPS attack exploits these structural traces—the skew in the signal probabilities—to locate and isolate the protection logic [26]. The attack becomes less effective in the presence of structural/functional obfuscation (refer to Sect. 7.1 for details).

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 [29] (refer to Sect. 7.2 for details).

Sensitization-Guided SAT (SGS) Attack The SGS attack exploits the security vulnerabilities of ATD to weaken the security it promises. The attack leverages the

bias in the input distribution of ATD to reduce the computational effort (refer to Sect. 7.3 for details).

Bypass Attack The Bypass attack adds *bypass* circuitry around a locked netlist [23] to restore its correct functionality. The attack assigns a random key to the locked netlist and determines the DIP(s) for which the netlist outputs are incorrect; the bypass circuit is then designed to restore the output for the computed DIPs (refer to Sect. 7.4 for details).

2.4.3 Side-Channel Attacks

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 [7]. 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 [24]. 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 [24]. The DPA attack may be launched on the individual logic cones in a divide-and-conquer approach (refer to Sect. 10.1 for details).

Test-Data Attacks While the regular attacks on logic locking rely on input/output observations recorded from a functional IC, test-data attacks, such as the hill climbing attack [12] and the test-data mining attack [28], 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 (refer to Sects. 10.2 and 10.3 for details).

De-synthesis Attack The de-synthesis attack relies on the observation that the locked and the original netlist should be similar in terms of the type and count of gates [11]. 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 (refer to Sect. 10.4 for details).

2.5 Attack-Defense Matrix

Table 2.1 presents the resiliency of various logic locking countermeasures against each of the attacks mentioned above. It can be observed, for example, that the SAT attack [21] breaks all Pre-SAT logic locking defenses. Compound logic locking that

Table 2.1 Attack resiliency of logic locking techniques

Attack	RLL [17]	FLL [1, 15]	SLL [14]	AntiSAT [22]	SARLock [25]	SFLL [30]	Compound [22, 25]	Cyclic [18]	ORF-Lock [27]
Sensitization [14]	✗	✗	✓	✓	✓	✓	✓	✓	✓
SAT [21]	✗	✗	✗	✓	✓	✓	✓	✓	✓
CycSAT [32]	✗	✗	✗	✓	✓	✓	✓	✗	✓
AppSAT [19]	✗	✗	✗	✓	✓	✓	✗*	✓	✓
Double-DIP [20]	✗	✗	✗	✓	✓	✓	✗*	✓	✓
Removal/SPS [26]	✓	✓	✓	✗	✗	✓	✓	✓	✗
AGR [29]	✓	✓	✓	✗	✗	✓	✗	✓	✓
Bypass [23]	✓	✓	✓	✗	✗	✓	✓	✓	✓
Test-data mining [28]	✗	✗	✗	✓	✓	✓	✓	✓	✓
Hill climbing [12]	✗	✗	✓	✓	✓	✓	✓	✓	✓
DPA [24]	✗	✗	✗	✓	✓	✓	✓	✓	✓
Desynthesis [11]	✗	✗	✗	✓	✓	✓	✓	✓	✓

✗ denotes that a technique is vulnerable the attack, ✓ denotes resilience against the attack, and ✗* denotes partial attack success

integrates point function-based locking with Pre-SAT defenses can be circumvented using the approximate attacks such as AppSAT or Double-DIP. 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.

2.6 The Ever-Evolving Metrics

The metrics used to evaluate the effectiveness of logic locking have also evolved along with the logic locking techniques. Many of the metrics quantify the security of a given defense against specific attacks. Following is a summary of the metrics deployed over the years:

- **Output corruptibility (OC)**. This metric quantifies protection against black-box usage of an IC; it was first used to highlight the advantage of FLL [15] over RLL [17]. An RLL netlist has the disadvantage that it may produce correct output for a large fraction of input/key combinations. Output corruptibility (OC) represents an estimate of the Hamming distance between the correct output and the incorrect output obtained on supplying random incorrect keys. The OC may be computed by applying P random input patterns, each with Q random key values, to the locked netlist L and observing its M-bit output O_L. OC represents the average percentage Hamming distance between the correct output O_F of the original function F and the output O_L obtained from the locked netlist.

  For a locked netlist L with

$$OC = \frac{1}{P \times Q \times M} \sum_{i=1}^{P} \sum_{j=1}^{Q} HD(O_F(I_i), O_L(I_i, K_j)) \times 100\% \tag{2.1}$$

  Here, I_i and K_j represent a random input pattern and a random key value, respectively. An OC value of 50% is ideal as it maximizes the ambiguity for an attacker [15].
- **Output error rate** (**OER**). Related to OC is the concept of OER. While OC represents the average percentage Hamming distance between the reference and the observed output, OER represents the percentage of input patterns that lead to an incorrect output. OER only considers if any of the M output bits is incorrect, whereas OC also takes into the number of incorrect output bits.

$$OER = \frac{1}{P} \sum_{i=1}^{P} a_i \times 100\% \tag{2.2}$$

  Where,

$$a_i = \begin{cases} 1, \text{if} \quad (\sum_{j=1}^{Q} HD(O_F(I_i), O_L(I_i, K_j))) \geq 1 \\ 0, \text{otherwise} \end{cases} \tag{2.3}$$

OER is often used to quantify the error injected by various point function-based techniques. The terms OC and OER have been used interchangeably in the literature.

- **Clique size**. Clique size quantifies the interference among the key gates and represents the resilience against the sensitization attack; the metric was introduced in the context of SLL [14]. Key gates that protect one another such that an individual key bit cannot be sensitized to primary outputs without knowing/controlling the values of remaining key bits form a clique. The number of key gates that form the largest clique in a circuit is referred to as clique size [27].
- **Number of distinguishing input patterns (#DIPs)**. Distinguishing input patterns are the special input patterns that are computed by the SAT attack and help refine the key search space [21]. In each iteration, the attack applies computes a DIP, which along with the corresponding output of the functional IC, helps distinguish the feasible key values from the infeasible ones. The computational effort of the attack can be represented in terms of the number of distinguishing input patterns required for a successful SAT attack.
- **Percentage of key bits recovered**. Certain attacks may succeed only partially and extract the values for only a subset of the key bits [12, 21, 24]. The success rate of such attacks is quantified in terms of the percentage of the key bits retrieved correctly. A powerful attack will recover a significant percentage of the key bits, enabling the attacker to brute-force on the remaining key bits.
- **Execution time**. The execution time of an attack can also be used to demonstrate the resilience of a logic locking technique against a specific attack [21]. Conventionally, the execution time is empirically recorded for smaller key sizes and then extrapolated to show the trend for larger key sizes [22, 25, 30].

To conclude, this chapter presented a summary of logic locking attacks, defenses, and the metrics used to evaluate logic locking algorithms. The chapter emphasized the relationship between the logic locking attacks and defenses in the form of an attack-defense matrix. It also presented a timeline of major developments in combinational logic locking. Overall, the chapter serves as a preview into the contents of the forthcoming chapters of the book.

References

1. Baumgarten A, Tyagi A, Zambreno J (2010) Preventing IC piracy using reconfigurable logic barriers. IEEE Des Test Comput 27(1):66–75
2. Chakraborty RS, Bhunia S (2009) HARPOON: an obfuscation-based SoC design methodology for hardware protection. IEEE Trans Comput Aided Des Integr Circuits Syst 28(10):1493–1502
3. Colombier B, Bossuet L, Hély D (2017) Centrality indicators for efficient and scalable logic masking. In: EEE computer society annual symposium on VLSI, pp 98–103
4. Dupuis S, Ba P, Natale GD, Flottes M, Rouzeyre B (2014) A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In: IEEE international on-line testing symposium, pp 49–54

5. Karousos N, Pexaras K, Karybali IG, Kalligeros E (2017) Weighted logic locking: a new approach for IC piracy protection. In: IEEE international symposium on on-line testing and robust system design, pp 221–226
6. Khaleghi S, Zhao KD, Rao W (2015) IC piracy prevention via design withholding and entanglement. In: Asia Pacific design automation conference, pp 821–826
7. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Advances in cryptology. Springer, Berlin, pp 388–397
8. Koushanfar F (2012) Provably secure active IC metering techniques for piracy avoidance and digital rights management. IEEE Trans Inf Forensics Secur 7(1):51–63
9. Lee YW, Touba N (2015) Improving logic obfuscation via logic cone analysis. In: Latin-American test symposium, pp 1–6
10. Li M, Shamsi K, Meade T, Zhao Z, Yu B, Jin Y, Pan D (2016) Provably secure camouflaging strategy for IC protection. In: IEEE/ACM international conference on computer-aided design, pp 28:1–28:8
11. Massad M, Zhang J, Garg S, Tripunitara M (2017) Logic locking for secure outsourced chip fabrication: a new attack and provably secure defense mechanism. CoRR abs/1703.10187, http://arxiv.org/abs/1703.10187
12. Plaza S, Markov I (2015) Solving the third-shift problem in IC piracy with test-aware logic locking. IEEE Trans Comput Aided Des Integr Circuits Syst 34(6):961–971
13. Rajendran J, Pino Y, Sinanoglu O, Karri R (2012) Logic encryption: a fault analysis perspective. In: Design, automation and test in Europe, pp 953–958
14. Rajendran J, Pino Y, Sinanoglu O, Karri R (2012) Security analysis of logic obfuscation. In: IEEE/ACM design automation conference, pp 83–89
15. Rajendran J, Zhang H, Zhang C, Rose G, Pino Y, Sinanoglu O, Karri R (2015) Fault analysis-based logic encryption. IEEE Trans Comput 64(2):410–424
16. Roy JA, Koushanfar F, Igor L (2008) EPIC: ending piracy of integrated circuits. In: Design, automation, and test in Europe, pp 1069–1074
17. Roy J, Koushanfar F, Markov IL (2010) Ending piracy of integrated circuits. IEEE Comput 43(10):30–38
18. Shamsi K, Li M, Meade T, Zhao Z, Pan DZ, Jin Y (2017) Cyclic obfuscation for creating sat-unresolvable circuits. In: ACM Great Lakes symposium on VLSI, pp 173–178
19. Shamsi K, Li M, Meade T, Zhao Z, Z D, Jin Y (2017) AppSAT: approximately deobfuscating integrated circuits. In: IEEE international symposium on hardware oriented security and trust, pp 95–100
20. Shen Y, Zhou H (2017) Double DIP: Re-Evaluating Security of Logic Encryption Algorithms. Cryptology ePrint Archive, Report 2017/290, http://eprint.iacr.org/2017/290
21. Subramanyan P, Ray S, Malik S (2015) Evaluating the security of logic encryption algorithms. In: IEEE international symposium on hardware oriented security and trust, pp 137–143
22. Xie Y, Srivastava A (2016) Mitigating SAT attack on logic locking. In: International conference on cryptographic hardware and embedded systems, pp 127–146
23. Xu X, Shakya B, Tehranipoor M, Forte D (2017) Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks. In: International conference on cryptographic hardware and embedded systems, pp 189–210
24. Yasin M, Mazumdar B, Ali SS, Sinanoglu O (2015) 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, pp 97–102
25. Yasin M, Mazumdar B, Rajendran J, Sinanoglu O (2016) SARLock: SAT attack resistant logic locking. In: IEEE international symposium on hardware oriented security and trust, pp 236–241
26. Yasin M, Mazumdar B, Sinanoglu O, Rajendran J (2016) Security analysis of anti-SAT. In: IEEE Asia and South Pacific design automation conference, pp 342–347
27. Yasin M, Rajendran J, Sinanoglu O, Karri R (2016) On improving the security of logic locking. IEEE Trans Comput Aided Des Integr Circuits Syst 35(9):1411–1424

28. Yasin M, Saeed SM, Rajendran J, Sinanoglu O (2016) Activation of logic encrypted chips: pre-test or post-test? In: Design, automation test in Europe, pp 139–144
29. Yasin M, Mazumdar B, Sinanoglu O, Rajendran J (2017) Removal attacks on logic locking and camouflaging techniques. IEEE Trans Emerg Top Comput. https://doi.org/10.1109/TETC.2017.2740364
30. Yasin M, Sengupta A, Nabeel MT, Ashraf M, Rajendran J, Sinanoglu O (2017) Provably-secure logic locking: from theory to practice. In: ACM/SIGSAC conference on computer & communications security, pp 1601–1618
31. Yasin M, Sengupta A, Schafer B, Makris Y, Sinanoglu O, Rajendran J (2017) What to lock?: functional and parametric locking. In: Great Lakes symposium on VLSI, pp 351–356
32. Zhou H, Jiang R, Kong S (2017) CycSAT: SAT-based attack on cyclic logic encryptions. In: IEEE/ACM international conference on computer-aided design, pp 49–56