Inter-Lock: Logic Encryption for Processor Cores Beyond Module Boundaries

Dominik Šišejković*, Farhad Merchant*, Rainer Leupers*, Gerd Ascheid* and Sascha Kegreiss†
*Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Germany
{sisejkovic, merchantf, leupers, ascheid}@ice.rwth-aachen.de
†Hensoldt Cyber GmbH, Germany
sascha.kegreiss@hensoldt.net

PREPRINT - published in ETS 2019, doi: 10.1109/ETS.2019.8791528

***Abstract*—The lack of technical resources and the high cost of establishing a semiconductor foundry has forced most integrated circuit design houses to rely on outsourcing part of their design and fabrication services to off-site companies. The involvement of external parties has given rise to major security threats, ranging from intellectual property piracy to the insertion of malicious circuits. Logic encryption has emerged as a popular mitigation technique against these threats. In recent years, a vast amount of logic encryption algorithms has been proposed. However, existing approaches strongly focus on isolated circuit components without taking the complexity and modular structure of modern circuit designs into account. In this paper, we propose Inter-Lock, a novel logic encryption framework tailored towards scaling logic encryption to larger designs by leveraging their complexity and exploiting multi-module interdependencies to exponentially enhance the security of sequential circuits. To showcase the applicability of the approach, we present an extensive evaluation on a real-life 32-bit RISC-V core together with the analysis of the security-cost trade-off. Our recommended Inter-Lock configuration of the core features a 1024-bit key and 100% functional corruption for 13.2% area overhead, less than 1% power overhead and 22.2% delay penalty in the worst case.**

I. Introduction

The immense price of establishing and maintaining a semiconductor foundry has forced most Integrated Circuit (IC) companies to operate fabless by offloading their design fabrication to off-site foundries. Since trust in foundries cannot be assured, this acts as an enabler of major security threats, such as reverse-engineering [1], Intellectual Property (IP) piracy [2] and counterfeiting [3]. These threats lead to tremendous revenue losses, as only counterfeiting results in an annual loss of about $100 billion of global revenue of legitimate electronics companies [4]. In recent years, another security threat has gained a lot of attention; the insertion of malicious circuits known as *hardware Trojans* [5]. The utilization of these malicious modifications can lead to serious repercussions such as data leakage or denial of service, particularly in the area of automotive, military and medical electronics [6]. As a reaction, a variety of countermeasures has been introduced, including logic encryption, watermarking [7] and IC camouflaging [8].

Logic encryption is a popular protection method based on the insertion of additional gates into a gate-level netlist to mask the original functionality and topology of the design [9]. These additional gates are referred to as *key gates*, as they are driven by a set of key inputs. If an incorrect key is set to the key inputs, the IC produces incorrect outputs for at least some input patterns. Otherwise, a correct key results in correct operation for all input patterns. The amount and distribution of the additional gates represent the trade-off between security and cost; more gates imply more area/power/delay penalties, but can lead to higher security against certain attacks.

A multitude of logic encryption algorithms have been proposed in recent years [9]. However, with the increasing number of available decryption attacks [10], [11], [12], [13], it has become a major challenge designing resilient logic encryption algorithms with favorable security properties.

Moreover, one important pattern can be observed; the existing proposals focus on securing *isolated combinational circuits* even if they are part of a larger design, or *sequential circuits treated as one isolated component* (e.g., a singular gate-level netlist). This has multiple major drawbacks. First, since all components are considered in isolation, the attack complexity relies on the security of the most vulnerable component, i.e., all components can be attacked independently. Secondly, the isolation enables a variety of attacks that can otherwise be prevented. Finally, modern designs typically consist of a multitude of isolated but interconnected components, that we refer to as *modules* (e.g., a decoder or controller in a processor design). Treating the complete design as one component disables the application of expert knowledge about the IP: some components might be more vulnerable to an attack than others, requiring dedicated security measures.

Contributions: In this paper, we propose the *Inter-Lock framework* for the exploitation of module interdependencies for securing sequential circuits with logic encryption. With Inter-Lock, we address the problem of *scaling the applicability of any logic encryption algorithm to more complex mutli-module hardware designs* (e.g., a processor core). The main contributions of the framework include:

- An interdependence-based encryption methodology for an *exponential* increase of security, rendering attacks on isolated modules infeasible. To the best of our knowledge, this is the *first proposal* focusing on an encryption framework for multi-module implementations.
- Resilience against prominent attacks as well as high security properties for all included modules.
- A security metric for the complete design based on the effective key length.
- An extensive analysis of the security-cost trade-off based on the evaluation on a RISC-V core. The case study offers configurations with an effective key length of 512 to 4096 bits, for variable area overheads of up to 40%.

II. Background and Related Work

Before diving into details, we present the basic background on logic encryption. We do not focus on algorithmic details of prior work, since Inter-Lock is a logic encryption *framework* and not a specific algorithm. The relevant attacks are described together with the resilience analysis in Section V.

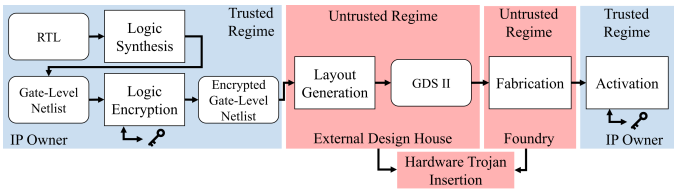


Fig. 1: Logic Encryption: IC Design and Fabrication Flow.

Logic Encryption in the IC Supply Chain: In the IC design and fabrication flow (Figure 1), it is often the case that the IP owner subcontracts parts of the design services, e.g., placement and routing, to an external design house due to the lack of necessary technical resources, tight deadlines and cost reduction. Afterwards, the generated layout is sent to the foundry. Both the external design house and the foundry are considered potentially malicious (untrusted regime), as they can insert a hardware Trojan into the design. To perform an effective hardware Trojan insertion, *the malicious entity has to fully understand the design*. This can be achieved through the analysis of the gate-level netlist, which can also be reverse-engineered from the layout [1]. To hinder this process, the IP owner can apply logic encryption to the gate-level netlist before the layout generation (trusted regime). After fabrication, the IC is returned to the IP owner for activation. The secret key is stored in a tamper-proof memory. Since only the IP owner knows the correct key, logic encryption hampers the ability to understand and reverse-engineer the design whilst being in hands of external entities.

Security Objectives of Logic Encryption: Regardless of the attack, logic encryption must provide two security properties: *functional* and *structural* corruption [14], [15]. Functional corruption describes the output corruptibility for incorrect keys; to hide the functionality of a locked circuit, it is favorable that incorrect keys generate corrupted (incorrect) outputs for as many input patterns as possible. Complementary to functional corruption, the structural corruption captures the corruption of the original circuit topology; the more the key gates are topologically distributed, the more structural difference is created between the original and encrypted circuit.

Logic Encryption Algorithms and Attacks: Earlier logic encryption algorithms focused on different security objectives, such as the correct location to insert a key gate or the desired circuit behavior for incorrect keys. The first proposal for hiding design details is known as EPIC [16]. This scheme is based on randomly inserting XOR/XNOR gates into the circuit. Later proposals include more complex strategies. For example, Strong Logic Locking (SLL) inserts key gates with a strong interference among them [17].

In the last years, many attacks to recover the correct key have been proposed [10], [17], [9]. The most successful among them is known as the Boolean Satisfiability (SAT) attack [11]. All previously known logic encryption algorithms have been shown to be vulnerable to this attack. Consequently, recently published algorithms strongly focus on mitigating SAT-based attacks [18]. Nevertheless, existing SAT-resilient approaches often suffer from low functional corruption (e.g., the majority of incorrect keys generate correct outputs) and leave structural traces vulnerable to removal-based attacks [10].

Recently, the key-interdependency logic encryption has been proposed [19]. Even though the name of the algorithm suggests a similarity to our work, it is fundamentally different from Inter-Lock. First, it focuses on the interdependency of internal key gates of an isolated module, while we utilize the interdependency of multiple modules. Secondly, the key-interdependency algorithm is a single specialized algorithm and not a framework. In fact, the algorithm can be used as one step of the Inter-Lock framework (see Section III).

The Security and Attack Model: The *security model* consists of the following assumptions: (i) the logic encryption key is secret and known only to the IP owner, (ii) fully reverse-engineering an encrypted design is only possible if the correct key is known and (iii) targeted hardware Trojan insertion is only feasible if the design is fully reverse-engineered and understood by the attacker.

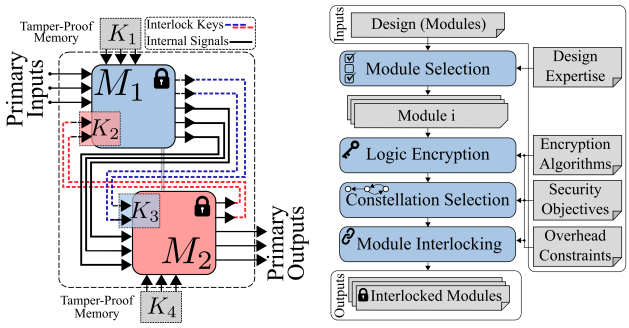
The *attack model* includes the following assumptions: (i) the attacker has access to a fabricated and activated IC to use as oracle, (ii) the attacker has access to the encrypted gate-level netlist and (iii) the location of the key inputs is known. The attacker can stimulate the primary and key inputs as well as trace the generated outputs, enabling a comparison to the input-output pairs of the activated IC. As the key inputs are known, the encryption scheme must assure that a simple detection and removal of the key gates is not possible.

III. The Inter-Lock Framework

Inter-Lock exploits the inherent multi-module nature of common hardware designs; different modules mutually depend on each other. This dependence is manifested through forward and backward connections (e.g., forwarding in pipelined processors). Furthermore, Inter-Lock *undermines one of the core assumptions* of logic encryption, namely that the attacker can always identify all key inputs due to their connection to the tamper-proof memory. If part of the key inputs cannot be identified, separately attacking isolated modules is rendered infeasible; e.g., the SAT attack on an isolated module cannot be performed as the exact location of the key inputs must be known. To achieve these goals, two or more modules are connected and their keys are made interdependent; *part of the key for each module is generated as part of the original functionality of other modules*. As forward/backward connections are very common in modern designs, key inputs cannot be differentiated from ordinary inputs/outputs.

A. A Motivational Example

The basic idea of Inter-Lock is presented in Figure 2 (a). The design consists of two connected modules M_1 and M_2. A part of the key inputs needed to activate both modules comes from the tamper-proof memory (bit vectors K_1 and K_4). The second part of the key for each module (bit vectors K_2 and K_3) is generated in the other module. These key connections between modules are dubbed *interlocks* and enable dependencies in both directions (M_1 to M_2 and vice versa). The input of the interlock is a key bit generated by the source module, while its output is connected to the input of the destination module. Therefore, each module acts as source and destination for a set of interlocks. Since the key bit generation becomes part of the functionality of a module, the interlocks are indistinguishable



(a) Module Interlocking Example (b) Inter-Lock Toolflow

Fig. 2: The Inter-Lock Framework.

from common internal signals. Consequently, the assumed attack model is not applicable any longer. To avoid unbuffered circular dependencies, the interlocks can be inserted in a cycle-preventive way or utilize registers (e.g., a pipeline). Moreover, the created interdependency yields an important result; an attacker cannot attack modules individually, hence *one has to consider all connected modules at once*. Therefore, the effective key length is exponentially increased.

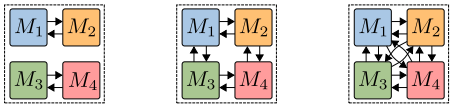
B. The Inter-Lock Toolflow

The complete Inter-Lock toolflow is presented in Figure 2 (b). It consists of four steps: (i) *module selection*, (ii) *application of logic encryption*, (iii) *constellation selection* and (iv) *module interlocking*. The input for Inter-Lock is a set of selected original circuit modules, while the output is a set of encrypted and additionally interlocked modules. All steps are discussed in detail in the following sections.

1) Module Selection: The input modules can be selected without any restrain. However, it is favorable to select modules containing critical functionality, such as the Arithmetic Logic Unit (ALU) or the decoder in a processor design. Such modules can be selected based on how likely they are to be utilized for, e.g., hardware Trojan insertion.

2) Application of Logic Encryption: The second step includes logically encrypting all selected modules. As with the module selection, one is free to choose whatever algorithm seems most beneficial. Because Inter-Lock disables attacking separate modules under the common attack model, one can focus on achieving security in terms of corruption instead of thwarting specific attacks (e.g., the SAT attack).

3) Constellation Selection: Constellations represent the possible structures in which the chosen modules are connected. Even though there is no limit to how a constellation can look like, some configurations are more beneficial than others. For evaluation purposes, in this work we focus on three fixed constellations: *pair*, *ring* and *network*. The pair constellation is the simplest one, connecting two modules in a two-way fashion (Figure 3 (a)). The ring constellation connects modules in a ring of neighbors; each module is connected with its two adjacent modules (Figure 3 (b)). The most complex network constellation connects all modules with each other (Figure 3 (c)). The incurred area overhead rises with the density of the connections (interdependencies); every interlock signal needs to be generated by an interlock circuitry.

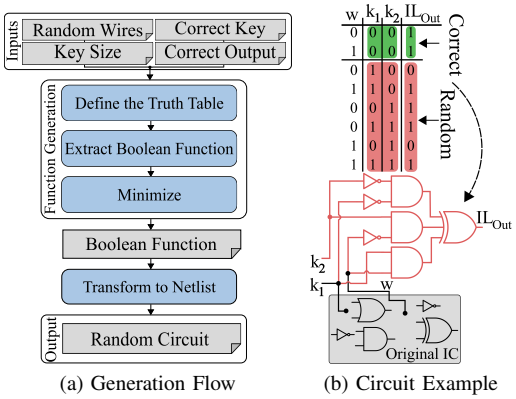


(a) Pair (b) Ring (c) Network

Fig. 3: Fixed Constellation Examples.

4) Module Interlocking: Interlocking refers to the insertion of logic for the generation of interlocks for other modules. Since one module must operate correctly when the correct key is applied, the generated output key bits must be correct and constant. A second important aspect is the stealthiness of the interlocking mechanism; *the interlocking circuitry must be indistinguishable from the rest of the circuit structure*. To assure constant interlock key bits, we propose the insertion of an additional encrypted circuitry that outputs the correct key when activated. The circuitry functions properly when the correct encryption key is provided. The key can be derived from the tamper-proof memory as well as depend on the interlocks generated in other modules. This has a beneficial effect; the interlocking and regular outputs have independent activation keys. Finding one key does not guarantee the correct operation of the whole design. Moreover, since the interlocking outputs cannot be identified, a correct separation of the total key is not feasible. After resynthesis, the interlocking circuitry becomes a regular part of the functionality of the original design. To assure removal-resilience, the additional circuitry must functionally and structurally depend on the original netlist. Therefore, we propose the random circuit method for the netlist generation of the interlocking circuitry.

The Random Circuit Method (RCM): RCM is based on the insertion of a random circuit that outputs the desired interlock key bit for the correct key input. The functionality of the circuit for incorrect keys can be freely defined. The RCM flow for one interlock key bit is shown in Figure 4 (a). The first step is to determine the input parameters, comprising: (i) the key length of the random circuit, (ii) a random wire, (iii) the correct key value and (iv) the correct output value. In the next step, a truth table is generated to describe the functionality of the circuit for correct and incorrect keys. From the truth table, the corresponding Boolean function is extracted, minimized and transformed into a gate-level netlist. Finally, the resulting circuit is embedded into the original netlist.



(a) Generation Flow (b) Circuit Example

Fig. 4: Random Circuit Method.

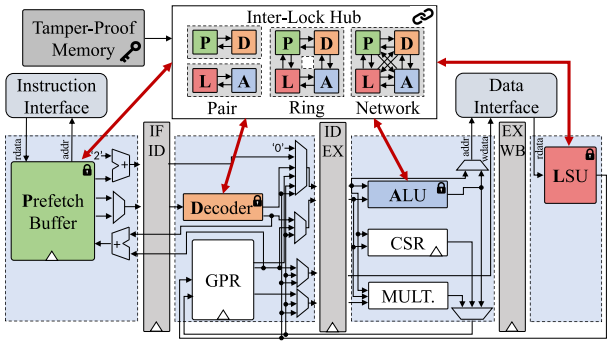


Fig. 5: Fixed Inter-Lock Constellations for the RI5CY Core.

One RCM configuration is illustrated in Figure 4 (b). The circuit depends on three wires: the regular key bit k_1 that acts as activation for part of the original functionality as well as the random circuit, a randomly selected regular wire w to increase the circuit obscurity, and the random circuit activation key bit k_2. Based on the selected behavior for incorrect keys (marked red), the flow generates a circuit that is inserted into the design to generate the value of the interlock IL_{Out}.

C. Security Metric

Let us assume that the security of one module is approximated with the lower bound key space of 2^{K_i}, where K_i is the key length of module i. This implies that the *effective key length* of a single module depends on how many other modules it depends on. Since Inter-Lock thwarts attacks on individual modules, the attacker must take the whole design (or at least all connected modules) into account. Therefore, for N connected (interlocked) modules, the security is represented with the effective key of length $K^e = \sum_{i=1}^{N} K_i$, i.e., a search space of size 2^{K^e}; representing an exponential increase compared to attacking each module individually. The attack on the overall design is still possible. However, now the design is considerably more protected due to the large amount of existing sequential elements (e.g., pipeline) and the exponential increase of the key length. Considering the amount of connected modules, the effective key length per fixed constellation is as follows: pair ($K^e = 2 \cdot K$), ring ($K^e = N \cdot K$) and network ($K^e = N \cdot K$). Note that ring and network achieve the same K^e as both connect all N modules.

IV. Experimental Evaluation

A. Experimental Environment

For evaluation, the PULPino System-on-Chip (SoC) was selected [20]. The SoC includes multiple SystemVerilog implementations of the open-source RISC-V instruction set architecture [21]. Logic synthesis was done with the Synopsys Design Compiler and the Faraday 90 nm library. ModelSim was used for Register-Transfer Level (RTL) and gate-level simulation. The framework was implemented in C++ and evaluated on an Intel i5 CPU@3.2 GHz with 8 GB of RAM.

B. RISC-V Case Study

Out of the available PULPino cores, the largest 32-bit RI5CY core was chosen for evaluation [20]. It features a four stage pipeline: Instruction Fetch (IF), Instruction Decode (ID), Execution (EX) and Write Back (WB). One module from each pipeline stage was chosen to be encrypted (Figure 5): the prefetch-buffer (IF stage), decoder (ID stage), ALU (EX stage) and the load-store unit (LSU, WB stage). These modules were chosen based on the assumption that these locations might be the easiest target for Trojan insertion as their functionality is well defined and crucial for the execution of all instruction types. For example, a simple Trojan can trigger a denial of service attack by blocking all further ALU operations after a specified sequence of instructions is detected in the decoder. We chose to encrypt all modules with EPIC [16], based on the insertion of XOR/XNOR gates at randomly chosen positions in the circuit. To avoid simple key gate removal, the gates are combined with inverters. Even though EPIC is vulnerable to the SAT and sensitization attack [11], due to the interdependent nature of Inter-Lock, the attacks are rendered infeasible; an exponentially large key space and the impossibility to detect key inputs make it extremely hard to perform an attack. Furthermore, the random spatial distribution of key gates in EPIC provides a maximal corruption of the original topology as well as a corrupted output for incorrect keys. However, since Inter-Lock is independent of the actual encryption, *any desired algorithm can be applied at this stage*.

TABLE I: Encrypted Module Size (Num. of Gates) for All Key Lengths. Values in Brackets Represent the % Area Increase.

Module	Original	256 Bit Key	512 Bit Key	1024 Bit Key
Decoder	1493	1876 (25.65%)	2260 (51.37%)	3028 (102.81%)
LSU	2383	2766 (16.07%)	3150 (32.19%)	3918 (64.41%)
Prefetch Buffer	8469	8852 (4.52%)	9236 (9.06%)	10004 (18.12%)
ALU	12652	13036 (3.04%)	13419 (6.06%)	14188 (12.14%)

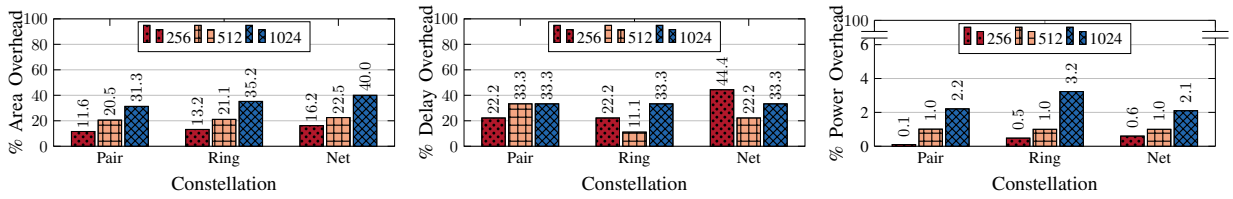
The selected modules were interlocked with RCM to form all three constellations for a per-module key length of 256, 512 and 1024 bits, with 16 interlock key bits between the modules. The constellations were configured as shown in Figure 5. To simplify the process of reincluding modules in the whole processor design, we generate an *Inter-Lock Hub*. This hub represents a SystemVerilog module with the goal to define the correct wiring between the selected modules. Once synthesized, the hub is integrated into the circuit structure.

The total amount of gates after encryption for each key length is shown in Table I. Depending on the module size, a key can imply different area increases (from 3.04% to 102.81%). In this work, we fixed the per-module key length to demonstrate the effect of encryption on the final module size.

C. Evaluation Results

ASIC Synthesis: The synthesis results are shown in Figure 6. The evaluation only considers the size of the core, not that of the entire platform, since only the core was modified during encryption. For the densest network constellation, the 256-bit, 512-bit and 1024-bit per-module configurations exhibit an area overhead of up to 16.2%, 22.5% and 40.0% respectively (Figure 6 (a)). Considering all key lengths, the results indicate that the cost of having denser connections (ring or network) is acceptable and should be prioritized.

The delay overhead is shown in Figure 6 (b). The fastest implementation for the original design has a critical path delay of 2.25 ns (Figure 7). Some constellations were not able to meet all tested timing constraints, as part of the encryption



(a) Area Overhead (b) Delay Overhead (c) Power Overhead

Fig. 6: RI5CY Core Overhead Increase for All Constellations and Per-Module Key Lengths (ASIC Synthesis).

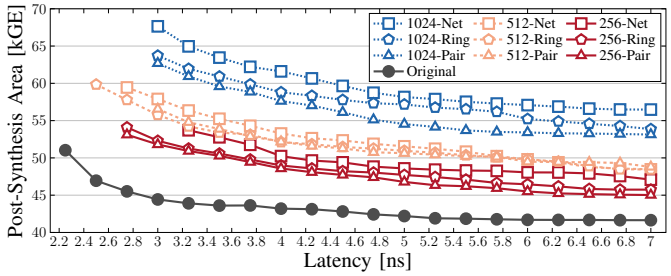


Fig. 7: RI5CY Core Post-Synthesis Latency-Area Graph for Different Constellations and Per-Module Key Lengths.

logic influences the critical path. The most complex network constellation and the 1024-bit per-module key yield a delay overhead of 33.3% (3.00 ns). Due to the randomness of EPIC, the delay degradation does not necessarily scale with the key length. For example, the highest delay overhead is generated for the 256-bit key and network constellation (44.4%).

The total power overhead is almost negligible with a maximum overhead of 3.2% (Figure 6 (c)). The reason for this is that the circuit power is dominated by the dynamic power consumption (more than 95%). As the interlocking and encryption circuitry cause no switching activity during correct operation, the total overhead remains very limited.

Since the security of Inter-Lock depends on the effective key length, it is favorable to connect as many modules as possible. However, denser connections yield no significant security improvement (except structural obfuscation), but imply a higher cost. Therefore, based on the previous analysis, we conclude that a ring constellation offers a good trade-off between cost and security, as it connects all modules with a minimum amount of interlocks. Furthermore, the total key length should be large enough to withstand brute-force attacks. Since the key lengths for multiple modules are accumulated, even a 256-bit key is sufficient. For example, the ring constellation for a per-module key length of $K = 256$ bits offers an effective key length of $K^e = 4 \cdot K = 1024$ bits. However, one must consider the corruption-per-module that a key implies, depending on the size of a module and the applied encryption algorithm.

Corruption Analysis: To approximate the functional and structural corruption, the per-module 256-bit key was chosen. The reason behind this choice is that the corruption rates for the smallest key offer a *lower bound* on corruption rates; a larger key is expected to yield higher corruption, especially for the EPIC encryption. The functional corruption was measured by recording the output for 10^6 input and incorrect key combinations and comparing it to the expected output for all selected modules individually. All modules result in a functional corruption rate of 100%; *an incorrect key generates incorrect outputs for all selected input patterns*.

The structural corruption is assured through the spatial distribution of key gates. The more the key gates are distributed in the circuit topology, the more structural difference between the original and encrypted circuit can be expected after resynthesis [14], [15]. The key gates and original gates are mapped to specific cells, embedding the encryption mechanism in the original circuit. Moreover, the interdependent nature of Inter-Lock assures high *corruption propagation*; an incorrectly set key bit in one module can corrupt not just the functionality, but also the activation of other modules.

Optimization Opportunities: To save area, the key length can be adapted to the initial module size based on the effective key length and the desired corruption. The number of interlocks can be adjusted to better fit each module depending on its structure and functionality in order to offer a more obfuscated design. The delay degradation can be hampered by avoiding the insertion of key gates into the critical path.

Implementation Remark: Due to the module interdependency, the keys in Inter-Lock need a few cycles to stabilize. For the PULPino SoC this requires no additional effort, since all code benchmarks start with a set of *nop* instructions.

V. Attack Resilience Analysis

In this section, we discuss the resilience of Inter-Lock against the relevant latest attacks [9].

SAT-Based Attacks: These attacks utilize a SAT solver to compute Distinguishing Input Patterns (DIPs) for which two key values produce different outputs [11]. The attack's efficiency lies in finding DIPs that can simultaneously eliminate multiple incorrect keys. Since Inter-Lock protects a selected number of key inputs for multiple modules, the attacker must first correctly guess *the position and the amount* of all key inputs (interlocks) *for every connected module* in the encrypted netlist. This becomes tremendously difficult due to the following reasons: (i) modules are inherently densely interconnected and contain a large number of inputs/outputs, (ii) the corruption propagation implied by Inter-Lock forces a high functional corruption rate for all modules, making it difficult to *isolate* known components through their expected functionality, and (iii) the design of interlocks makes them indistinguishable from ordinary signals. Moreover, treating interlocks as regular inputs will necessarily yield an incorrect key for each module (equivalent to ignoring a part of a key). Therefore, the attacker has to apply the attack on all modules simultaneously. However, executing a SAT attack on sequential ICs is only possible if full access to the scan chain

is granted to control internal circuit states [22]. Thereby, the scan chain can simply be secured through various existing solutions [23]. However, a SAT attack is still possible on IC camouflaging [22]. Since camouflaging adapts the layout of selected cells [24], it falls into the untrusted regime (see Section II) and is therefore beyond the scope of this work.

Sensitization Attack: The sensitization attack is based on computing a dedicated input pattern to sensitize a key value in the activated IC to a primary output [17]. As before, to fully apply this attack, the attacker must first identify all key inputs for every module and have full access to the scan chain. For additional per-module resilience, the SLL encryption can be utilized as one possible algorithm in the Inter-Lock framework to secure against sensitization attacks [17].

Test-Data Attacks: Instead of using an activated IC as oracle, test-data attacks extract secret data about the key from the test data. For example, the Hill Climbing attack [25] tries to achieve a zero Hamming distance between the encrypted circuit and the test response for random key guesses. However, it has been shown that test-data attacks can be circumvented with, e.g., SLL (applicable with Inter-Lock) and post-test activation [26] (independent of logic encryption).

Structural Analysis Attacks: An attacker can try to structurally analyze the netlist to identify all components and anticipate the amount of original inputs to rule out the interlocks. However, the attacker must first correctly identify and isolate every connected module through its functionality or structure. Both aspects are protected through selected encryption algorithms, a suitable number of interlocks, corruption propagation and the indistinguishability of the interlocking circuitry.

Differential Power Analysis (DPA): This attack statistically correlates the power trace and the input data of an IC to determine the correct key bits [13]. Within Inter-Lock, DPA can be thwarter by using DPA-resistant logic encryption [9]. Furthermore, the applicability of DPA still remains to be evaluated for more realistic logic encryption configurations; so far, it was evaluated only for 32-bit logic encryption keys [13]. Note that storing a large encryption key is of no concern.

Desynthesis Attack: This attack works as follows. First, the encrytped netlist is resynthesized with a random key. Afterwards, the attack utilizes a Hill Climbing search to find a key that yields a maximum similarity between the encryted and resynthesized netlist. In the context of Inter-Lock, this attack can be prevented by using desynthesis-resilient logic encryption [9]. Furthermore, the attack assumes that all key input locations are known (not true for Inter-Lock) and the applicability of the attack has been evaluated only for benchmarks with 32/64-bit keys [12].

Algorithm-Specific Attacks: Some attacks are applicable only to very specific logic encryption algorithms by utilizing algorithm-specific features. These include the Signal Probability Skew (SPS), the AppSAT Guided Removal (AGR) and the bypass attack [10]. In Inter-Lock, all can be circumvented by selecting resilient logic encryption algorithms [9], [10].

VI. Conclusion

In this paper, we presented Inter-Lock; a novel framework for scaling logic encryption to multi-module hardware designs. By exploiting inter-module dependencies, Inter-Lock disables attacks on separate modules, offering an exponential increase of security and an effective key length security metric. The framework was evaluated on a 32-bit RISC-V processor core for an ASIC synthesis flow. The results were analyzed regarding the trade-off between area/power/delay penalties and security. The case study offers an effective key length of 512 to 4096 bits for area overheads from less than 12% to 40%. The results show that Inter-Lock is able to achieve both the resilience against prominent attacks and high structural and functional corruption. With Inter-Lock, we encourage more research on developing effective encryption frameworks by leveraging the complexity of modern hardware designs. In future work, we plan to update the framework in regards to the mentioned optimization opportunities.

References

[1] M. Fyrbiak *et al.*, "Hardware reverse engineering: Overview and open challenges," in *IVSW 2017*, July 2017, pp. 88–94.

[2] M. Rostami *et al.*, "A primer on hardware security: Models, methods, and metrics," *Proc. of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug 2014.

[3] U. Guin *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. of the IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug 2014.

[4] M. Pecht and S. Tiku, "Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spec.*

[5] K. Xiao *et al.*, "Hardware trojans: Lessons learned after one decade of research," vol. 22, pp. 1–23, 05 2016.

[6] S. Bhunia *et al.*, "Hardware trojan attacks: Threat analysis and countermeasures," *Proc. of the IEEE*.

[7] R. D. Newbould *et al.*, "Watermarking ICs for IP protection," *Electronics Letters*, vol. 38, no. 6, pp. 272–274, Mar 2002.

[8] R. P. Cocchi *et al.*, "Circuit camouflage integration for hardware IP protection," in *DAC 2014*, June 2014, pp. 1–5.

[9] M. Yasin and O. Sinanoglu, "Evolution of logic locking," in *2017 IFIP/IEEE VLSI-SoC*, Oct 2017, pp. 1–6.

[10] M. Yasin *et al.*, "Removal attacks on logic locking and camouflaging techniques," *IEEE TETC*, vol. PP, no. 99, pp. 1–1, 2017.

[11] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *IEEE HOST 2015*, May 2015, pp. 137–143.

[12] M. E. Massad *et al.*, "Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism," *arXiv preprint arXiv:1703.10187*, 2017.

[13] M. Yasin *et al.*, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *2015 IEEE DFTS*, Oct 2015, pp. 97–102.

[14] H. Zhou, "A humble theory and application for logic encryption," *IACR Cryptology ePrint Archive*, vol. 2017, p. 696, 2017.

[15] D. Šišejković *et al.*, "A unifying logic encryption security metric," in *2018 SAMOS*, Jul 2018.

[16] J. A. Roy *et al.*, "EPIC: Ending piracy of integrated circuits," in *2008 DATE*.

[17] M. Yasin *et al.*, "On improving the security of logic locking," *IEEE TCAD*, vol. 35, no. 9, pp. 1411–1424, Sept 2016.

[18] M. Yasin *et al.*, "Provably-secure logic locking: From theory to practice," in *2017 ACM SIGSAC*. ACM, 2017, pp. 1601–1618.

[19] R. Karmakar *et al.*, "A new logic encryption strategy ensuring key interdependency," in *2017 30th VLSID*, Jan 2017, pp. 429–434.

[20] A. Traber *et al.*, "Pulpino: A small single-core RISC-V SoC," in *3rd RISCV Workshop*, 2016.

[21] A. Waterman *et al.*, "The RISC-V instruction set manual," *volume I: User-level ISA, version 2.0, Tech. Rep. UCB/EECS-2014-54*, 2014.

[22] M. E. Massad *et al.*, "Reverse engineering camouflaged sequential circuits without scan access," in *ICCAD '17*. Piscataway, NJ, USA: IEEE Press, 2017, pp. 33–40.

[23] J. Lee *et al.*, "A low-cost solution for protecting IPs against scan-based side-channel attacks," in *24th IEEE VTS*, April 2006, pp. 6 pp.–99.

[24] M. Yasin and O. Sinanoglu, "Transforming between logic locking and IC camouflaging," in *2015 IDT*, Dec 2015, pp. 1–4.

[25] S. M. Plaza and I. L. Markov, "Protecting integrated circuits from piracy with test-aware logic locking," in *2014 ICCAD*, Nov 2014, pp. 262–269.

[26] M. Yasin *et al.*, "Activation of logic encrypted chips: Pre-test or post-test?" in *2016 DATE*, March 2016, pp. 139–144.