Revealing On-chip Proprietary Security Functions with Scan Side Channel Based Reverse Engineering

Leonid Azriel, Ran Ginosar and Avi Mendelson
Viterbi Faculty of Electrical Engineering
Technion – Israel Institute of Technology, Haifa 32000 ISRAEL
leonida@tx.technion.ac.il, ran@ee.technion.ac.il, avi.mendelson@technion.ac.il

ABSTRACT

Proprietary cryptographic algorithms or protection schemes often constitute part of the security solution in electronic devices. Hence, these devices are prone to reverse engineering attacks that may reveal the details of these algorithms. We propose a novel non-invasive method of reverse engineering of digital integrated circuits that exploits the scan chains originally inserted into the device for production test automation. The scan chains unfold the sequential logic of the device to form a combinational function. The device's functionality is then exposed by examining this function. The resulting function is too large for direct learning, so we developed heuristic learning algorithms that exploit common properties of digital circuits, in particular limited transitive fan-in of combinational logic and sub-circuit sharing properties. With these algorithms we show fast reconstruction of an AES cryptographic accelerator. The algorithm used for the AES is scalable, making it applicable to much larger circuits.

Keywords

Hardware reverse engineering, Side-channel analysis and countermeasures, Embedded Systems Security

1. INTRODUCTION

Reverse engineering of an integrated circuit is a complex task that traditionally requires tedious work and expensive equipment. Its ultimate goal is to discover the underlying algorithm of a given physical device: the device's behavioral definition. We can represent the discovery task as a two-stage process: (1) Extracting of the circuit from the physical device and (2) extracting of the behavioral model from the circuit. The boundary between the two stages may be blurred; nevertheless, these are usually two distinct tasks. The first stage, as a rule, involves invasive techniques, such as removing the package, performing cross-section, delayering, and nanoscale imaging [19]. The second stage is usually algorithmic [13, 15]. This paper addresses the first stage, circuit extraction, but proposes to do it non-invasively by exploiting the internal scan chains. This approach provides notable accuracy with simple and inexpensive equipment.

Scan insertion is a widely used DFT (Design-For-Test) technique that allows for the automatic generation of test vectors for production test of an integrated circuit. Thanks to its efficiency, it has become a de facto standard for testing digital circuits. Recent research shows that scan insertion may turn into an easy target for the attacker [6, 10]. Previous work primarily focuses on the vulnerability of the registers that participate in the scan chain. Easy access to these registers may reveal confidential information, such as cryptographic keys. This paper presents a different attack that reveals the functionality of the design itself. This security breach broadens the scope of possible exploits of the scan mode to any device that may contain trade secrets or proprietary cryptographic algorithms or protection schemes. The proposed reverse engineering method can also serve positive purposes, such as competitive analysis, IP theft detection [1], or discovery of malicious hardware implanted by a third party during physical design.

The scan introduces a new *shift mode*, which arranges the internal registers in one or more shift registers, called scan chains (see Figure 1). The production tester may switch the chip to the shift mode and use the scan chains to place the chip in the desired state ($ShiftIn$) and to sample its current state ($ShiftOut$). The $ShiftIn$ and $ShiftOut$ operations can be combined with a single functional ($Capture$) cycle to learn ($Probe$) the output of the combinational function F for a given input. An exhaustive search over all possible states of the device's registers and input pins then reveals a truth table that fully describes the function F.

The exhaustive search, due to its exponential complexity, is practical only for very small devices, containing no more

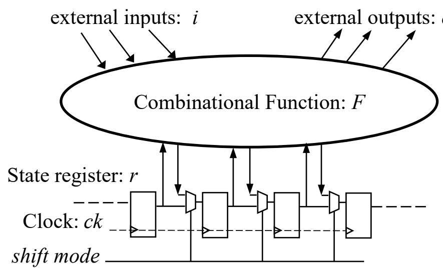

Figure 1: Scan Design.

GLSVLSI '17, May 10-12, 2017, Banff, AB, Canada
© 2017 ACM. ISBN 978-1-4503-4972-7/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3060403.3060464

than a few dozen registers. For the general case of learning an n-input Boolean function, the number of possible functions is 2^{2^n}. However, not every Boolean function can be realized by a physically constrained integrated circuit. In fact, most of the functions are not realizable. In other words, the search space in the case of integrated circuits is much smaller than in a general case. This is a direct corollary from the Shannon Effect [14]. Whether it is possible to devise an upper bound for the number of n-input Boolean functions with limited complexity is an open question. If such a bound exists, it may define the search space for our problem. Until this question is answered, we take a heuristic approach based on function properties. Our work correlates with the field of Computational Learning theory, which studies learning algorithms for functions with certain properties, such as linear functions, decision trees, DNF and junta functions [16]. We use some of the algorithms developed in this field, in particular algorithms that learn juntas.

We assume that the scan test interface is present and accessible in the target device. Vendors of secure integrated circuits often protect their scan interface with authentication, obfuscation or other mechanisms. The scan-based reverse engineering method may overcome some protection mechanisms, especially when combined with other methods, as detailed in Section 4. We refer the reader to [2] for more details and discussion.

2. ALGORITHMS FOR LEARNING A BOOLEAN FUNCTION WITH PROBES

The scan based attack turns the problem of reverse engineering into the problem of learning a stateless Boolean function. Here we describe the algorithms that learn the circuit's combinational function F exposed by the scan. We start with a formal definition of the circuit (see Figure 1).

Definition 1. *Let S be a digital circuit comprising a vector of inputs $i = (i_1, \ldots, i_a) \in \{0,1\}^a$, a vector of outputs $o = (o_1, \ldots, o_b) \in \{0,1\}^b$, a state register $r = (r_1, \ldots, r_n) \in \{0,1\}^n$, a clock input $ck \in \{0,1\}$ and a collection of combinational gates that implement the next state and output function $F = (F_0, \ldots, F_{n+b})$*
s.t. $(r \parallel o)_{next\_ck\_cycle} = F(r \parallel i)$, when $|r \parallel i| = N = n + a$.

The scan insertion arranges the bits of the state register r in c scan chains For brevity, we omit the description of the scan control, and assume immediate access to the state register r. However, for computing the time complexity, we keep in mind that the *Probe* operation lasts $2n/c+1$ clock cycles, one cycle for *Capture* and n cycles for each of the *ShiftIn* and *ShiftOut* operations. The parameter c depends on the number of pins, and we assume $c \ll n$. A probe operation $Probe(S, v)$ over circuit S is defined in Algorithm 1.

Algorithm 1 Probe(Circuit S, vector v)
```
1: $r \parallel i := v$          ▷ Set registers and inputs state to $v$
2: $o_{t-1} := o$                ▷ Sample outputs of $S$
3: Capture
4: **return** $r \| o_{t-1}$     ▷ New register values and outputs
```

A truth table of F can be obtained by running *Probe* operations on circuit S with all possible input vectors. This is an exhaustive search over a truth table algorithm (*ESoTT*) that grows exponentially and is not practical. Later in this section, we present algorithms that leverage the limited transitive fan-in property of the circuit S to reduce the learning complexity. Transitive fan-in reflects the number of inputs of a combinational logic cone; for a typical combinational logic cone, it is much smaller than N. Figure 2 shows a histogram of transitive fan-ins of the logic cones from the circuits in the ITC'99 benchmark [4] set. More than half of the cones have a transitive fan-in of 64 or smaller. This observation allows the use of heuristic based algorithms.

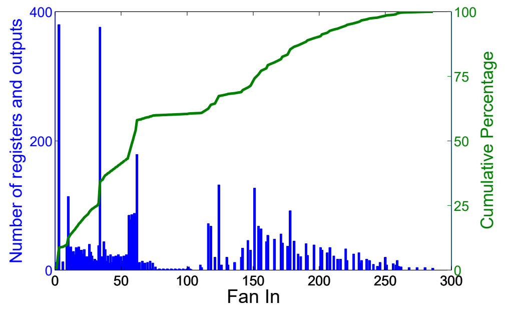

Figure 2: Transitive fan-in histogram for flip-flop inputs and primary outputs in the ITC'99 benchmark circuits.

2.1 Algorithm CSoTT (Compact K-Search over Truth Table)

In the Boolean function domain, the transitive fan-in is represented by the support size of the function. Strictly speaking, the transitive fan-in of a logic cone is greater or equal to the support size of the corresponding function. The learner may assume some limit K_{max} on the transitive fan-in of the combinational logic cones in the circuit of interest based on prior knowledge or statistics. Choosing $K_{max} \ll N$ reduces the search space significantly.

Every output bit y_i depends on some subset G_i of all the function variables. Hence, it is sufficient to examine the function F with a set of input vectors that cover all value combinations of all subgroups G s.t. $|G| \leq K_{max}$. As a trivial example, when $K_{max} = 1$, the function can be learned using the 0 vector and all input vectors, in which only one bit is set. More generally, to reconstruct the function F, it is sufficient to test it with a group of vectors with Hamming weight (HW) of up to K_{max}. *CSoTT* (Algorithm 2) implements this method by iterative probing of the function F with vectors v, starting with vectors with HW=0 and going up to K_{max}. If bit i of the probe result is equal to 1, i's entry of the *Onset* table is updated with a new implicant candidate, unless a cover of the corresponding term already exists in the table. If the bit is 0, but the *Onset* table includes an implicant candidate t, which is a cover of v, then t is removed from the table. Instead, *Onset* is updated with all the terms covered by t, all literals of which match bits in v equal to 1, excluding v itself.

For intuition, suppose that for some output bit i, the function F_i is positive unate, namely, all the literals in its DNF form are positive. Recording only positive literals is then sufficient to represent F_i. Moreover, only primary implicants can be stored. Now suppose there is a term t with a single negative and l positive literals. After l Hamming

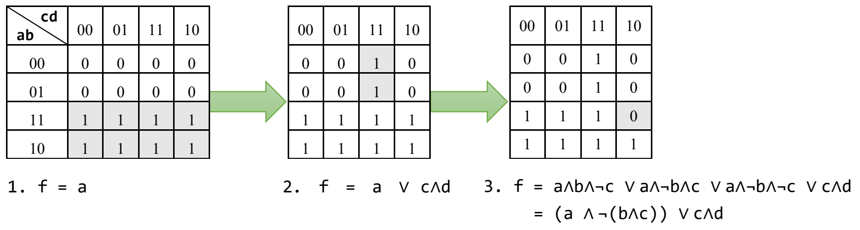

Figure 3: ***CSoTT*** **algorithm stages with their respective Karnaugh maps.**

weight iterations, the algorithm's table will contain a vector that includes all the positive literals in t. At the iteration $l+1$, the probe operation on the vector with all the variables from the term t set to 1 will return 0. As a result, the algorithm will add the required negative literal to the stored vector. This approach can be extended to any number of negative literals.

Algorithm 2 CSoTT

```
 1: $F0$ := Probe($S$,0)
 2: Onset[$i$] = $\emptyset$ for all $i$ from 1 to $n$+$b$
 3: for $K$ from 1 to $K_{max}$ do
 4:     for all $v \in \{0,1\}^N$ s.t. $HW(v)$=$K$ do
 5:         $P$ := Probe($S$,$v$)$\oplus F0$
 6:         $T$ := $\wedge T_j$ s.t. $v_j$=1                ▷ compose term
 7:         for $i$ from 1 to $n$+$b$ do
 8:             if $P[i]$=1 & cover($T$)$\notin$Onset[$i$] then
 9:                 add(Onset[$i$],$T$)
10:             else if $P[i]$ = 0 then
11:                 call updateOnset($T$,$i$)
12:             end if
13:         end for
14:     end for
15: end for
16: procedure UPDATEONSET($T$,$i$)
17:     for all $t \in$Onset[$i$]=cover($T$) do
18:         for all $\hat{t}$=cover($v$) s.t. $t$=cover($\hat{t}$) do
19:             $t_{inv}$ := $\wedge\neg t_n$ for $n$ s.t. $t_n$=1 AND $t_n \notin \hat{t}$
20:             $\tilde{t}$ := $\hat{t} \parallel t_{inv}$
21:             add(Onset[$i$],$\tilde{t}$)
22:         end for
23:     end for
24:     remove(Onset[$i$],$t$)
25: end procedure
```

The algorithm's iterative operation can also be demonstrated using Karnaugh maps. Figure 3 shows the algorithm stages for an example single-bit function $F(a,b,c,d) = (a \wedge \neg(b \wedge c)) \vee c \wedge d$. At stage 1 ($HW = 1$), the only recorded implicant candidate is a. At stage 2 ($HW = 2$) the implicant $c \wedge d$ is added. At stage 3 ($HW = 3$), the implicant candidate a is found incomplete, since the probe operation on a vector $\{a,b,c,d\} = 1110$ yields a result of 0. Thus, it is replaced with higher order terms as shown in the corresponding Karnaugh map in the figure. At the end of the algorithm's execution, the Onset table contains a DNF representation of the function F.

CSoTT Time and Space Complexity The number of probes in *CSoTT* equals the number of elements in $\{0,1\}^N$ with Hamming weight of K_{max} or smaller, that is $\sum_{i=0}^{K_{max}} \binom{N}{i} \leq 1 + N^{K_{max}}$. Hence, the time complexity of the algorithm can be written as:

$$T_{CSoTT} = O[(2n/c+1) \cdot (1 + N^{K_{max}})] \qquad (1)$$

The space required for *CSoTT* is bounded by the worst-case size of a DNF representation of a function with K_{max} variables. In practice, the memory space for *CSoTT* is roughly the size of the minimal DNF representation of F.

$$S_{CSoTT} = O[N_o \cdot K_{max} \cdot 2^{K_{max}-1}] \qquad (2)$$

Clearly, due to exponential growth, K_{max} of a typical circuit is too high for the algorithm to be practical. A lower K can be selected for partial reconstruction of the function F. For example, in approximately half of the cases from the ITC'99 statistics, the transitive fan-in is smaller than 50, and for 25% of them it is below 32. This partial information may provide an adversary with sufficient information to carry out the attack.

2.2 Algorithm ISoTT (Incremental CSoTT)

Algorithm 3 ISoTT

```
 1: Pick $K_{init}$, $K_{step}$
 2: $K$ := $K_{init}$
 3: Run CSoTT($K_{max}$=$K$) on $S$
 4: do
 5:     for all $m \in$Onset s.t. HW($m$)=$K_{init}$ do
 6:         for all $j$ s.t. $m_j \in m$: set $v_j$ to constant 1;
 7:         Run CSoTT($K_{max}$=$K$) on remaining bits of $v$
 8:     end for
 9:     $K$ := $K_{max}$ + $K_{step}$
10: while there is a change in Onset
```

ISoTT is a speculative algorithm that invokes *CSoTT* a number of times. After the first invocation of *CSoTT* with a computationally reasonable K_{max}, the *Onset* table will contain a set of implicants $\{T\}$, each comprising a set of variables $\{G_T\}$. We speculate that if the function F contains higher order terms, many of them are a superset of some set in $\{G_T\}$. This is supported by the observation that the same sub-circuit may belong to several logical cones. One example of this phenomenon is the carry propagation logic in arithmetic circuits. The *ISoTT* algorithm thus adds incremental phases, in which for each of the terms recorded in the *Onset* table, *CSoTT* runs over a subset of vectors, where each of the vectors is an extension of the vector represented by this term. The parameter K_{step} controls the magnitude of the extension. The algorithm runs as long as there is a change in the *Onset* table.

ISoTT implements a greedy best-first search method. At every step of the algorithm, *CSoTT* is called as many times as the number of implicants with the maximum Hamming weight in the table at the beginning of running this step. Thus, the time complexity of a single algorithm is:

$$T_{ISoTT} \le O[S_{ISoTT(step-1)} \cdot T_{CSoTT}(K_{max} = K_{step})] \quad (3)$$

From (3), $ISoTT$'s performance depends on the structure of the circuit and on the size of its minimal DNF representation. $ISoTT$ is based on speculation, therefore it does not guarantee correctness. However, for some examples, such as the arithmetic circuits, $ISoTT$ achieves full reconstruction.

2.3 Algorithm JSoTT (Junta based Search over Truth Table)

$JSoTT$ is a randomized algorithm that also exploits the limited transitive fan-in property. A function f: $\{0,1\}^n \rightarrow \{0,1\}$ is called a *k-junta* for $k \in x$ if it depends on at most k of its input coordinates; i.e., $f(x) = g(x_{i_1}, \cdots, x_{i_k})$ for some g: $\{0,1\}^n \rightarrow \{0,1\}$ and $i_1, \cdots, i_k \in [n]$ [16]. Hence, algorithms that learn junta functions from queries can be used to reconstruct combinational logic cones with a limited transitive fan-in. We take the adaptive algorithm from [7]. The algorithm runs for every output bit and comprises two phases: finding dependencies and function discovery. At the first phase, a set of probes with random input vectors is prepared. The results of the probes are used to find input bits affecting the output (relevant variables or RV).

Algorithm 4 JSoTT

```
 1: Phase 1: Find relevant variables
 2: $F0$ := Probe($S$,0)
 3: RV[$i$] = $\emptyset$ for all $i$ from 1 to $n$+$b$
 4: loop $K_{max} \cdot 2^{K_{max}}$ times
 5:     $v$ := random(1,...,$2^N - 1$)
 6:     $P$ := Probe($S$,$v$)$\oplus F0$
 7:     add(Probes, $\langle v,P \rangle$)
 8: end loop
 9: for $i$ from 1 to $n$+$b$ do        ▷ Each output bit separately
10:     for all $\langle v,P \rangle$ in Probes do
11:         $\hat{v} := \{\hat{v}_1, \ldots, \hat{v}_N\} = v$
12:         For all j from 1 to N: $\hat{v}_j = 0$ if ($v_j \notin$RV[$i$])
13:         $\hat{P}$ := Probe($S$,$\hat{v}$)$\oplus F0$
14:         if $P_i \neq \hat{P}_i$ then
15:             find next RV by binary search on $v$ keeping
    all $v_j \in RV[i]$ fixed[1]
16:             add(RV[$i$], RV)
17:         end if
18:     end for
19: end for
20: Phase 2: Discover logical functions
21: for $i$ from 1 to $n$+$b$ do
22:     Call CSoTT[2] for output bit $i$ with input composed
    from bits in RV[$i$] and $K_{max} = |RV[i]|$
23: end for
```

The algorithm finds variables that result in different probe results between vectors. Consider two input vectors v and u from the random vector set, such that $Probe(S,v) \neq Probe(S,u)$. The vectors v and u differ in the number of bits. If we start swapping the discriminating bits in v one by one (moving towards u), there will be a step at which the probe result will change. The variable that caused this change is a relevant variable. This operation is repeated for all the vectors in the set in order to find additional relevant variables, while the variables already known to be relevant will not be swapped. Finally, for faster search, rather than going one by one, the variables can be swapped in groups using the binary search principle. After the relevant variables are found for each output bit, a logical function is discovered by checking all the value combinations of the variables relevant to this output bit. This is the second phase.

JSoTT Time and Space Complexity. From [7]:

$$T_{JSoTT} \le O[K_{max} \cdot 2^{K_{max}} \cdot logN_o + N_o \cdot (logN_o + 2^{K_{max}})] \quad (4)$$

$JSoTT$ scales well with the circuit size. Hence, it is better than $ISoTT$ for large circuits. However, thanks to its heuristic nature, $ISoTT$ overperforms for certain circuits.

3. EXPERIMENTAL RESULTS

We evaluated the algorithms from Section 2 on a set of synthetic benchmarks and one real application benchmark. For this, we built a software simulator that models digital circuits with a $Probe$ operation: an operation that represents the circuit's next state function F according to Definition 1. The simulator also implements the algorithms defined in Section 2. The output of the algorithm, the circuit hypothesis function f, is further matched against the original circuit function F. The simulator performs the matching by comparing the outputs of the functions F and f using a statistical method with a sufficiently large sample set of randomly selected inputs. $ESoTT$ serves as a baseline. For $ISoTT$, we perform a series of runs with different K_{init} and K_{step} parameters, and then select the best results.

3.1 Validation of Concept

We first checked the correctness of the scan based implementation of $Probe$, with the Validation of Concept experiment. For this experiment, we used a simple incrementor circuit S. The circuit description was written in Verilog and synthesized, including scan insertion, using Synopsys tools. On the netlist with scan we ran $ESoTT$ using the Verilog behavioral simulator. Note that the algorithm execution was preceded by learning the length of the scan chain from shifting in a pattern to the scan input and counting the number of cycles until the pattern was observed at the scan output. The reconstructed circuit S' (Figure 4) comprises (1) a register R comprising all n flip-flops in the scan chain, and (2) a combinational function (represented by a truth table) obtained by $ESoTT$. After obtaining the reconstructed circuit S', we confirmed its equivalence to the original circuit S with a formal logic equivalence tool. The remainder of this section presents results from the software simulator, which uses the $Probe$ abstraction assuming correctness of its underlying scan-based implementation.

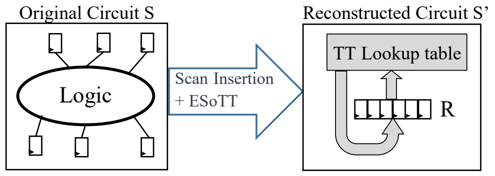

Figure 4: Validation of Concept Circuit Diagram.

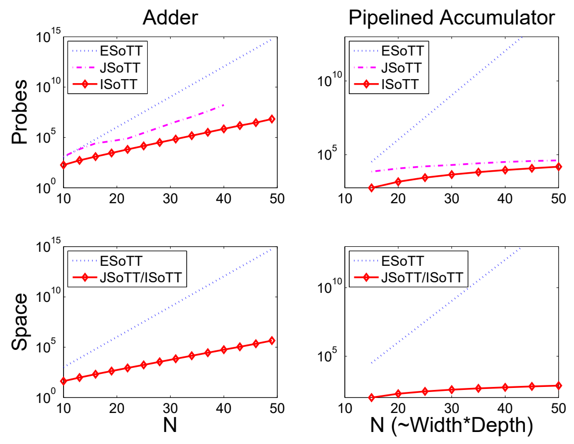

Figure 5: Algorithm results with arithmetic circuits. Runtime (in number of probes) and space requirements for a single adder and a pipelined accumulator with different sizes.

3.2 Arithmetic and Data Path Elements

We first evaluate the algorithms when applied to common building blocks of the digital circuits. We start with arithmetic circuits and measure the runtime and space required for a full reconstruction of the adder circuit. Although arithmetic circuits are characterized by tight dependency, which makes the limited fan-in optimization inefficient, their regular and recursive structure is useful for the incremental $ISoTT$ algorithm. Indeed, as shown in Figure 5, $ISoTT$ is the most efficient for the adder.

The adder implements a single function, where the limited fan-in approach has little advantage; thus, all the algorithms still run in exponential time. We expect to see the speedup with structures having loose dependencies between their sub-structures. As an example, we took a pipelined accumulator circuit built of pipeline stages; each merely adds the result of the previous stage to the input vector. When unfolded to a combinational structure, it turns to a set of adders in a parallel construction. Here, K_{max} is derived from the size of the single adder and does not depend on the number of pipeline stages. Therefore, we observe polynomial growth of space and time in Figure 5.

Next, we evaluate our algorithms with a multiplexer, a typical element of a data path. For example, unfolding a register file with scan results in a multiplexer. The size of the input vector to the multiplexer with bus width W and rank (number of input busses) R is $N{=}R{\cdot}W$. However, every output bit depends on only $R{+}log_2 R$ bits; namely, K_{max} does not depend on the bus width. This makes the algorithms, which leverage the limited fan-in, particularly efficient for wide data path structures. Interestingly, $ISoTT$ performs better for all the arithmetic circuits, despite $JSoTT$'s lower theoretical complexity. A partial explanation is that while for $JSoTT$ K_{max} represents the maximum transitive fan-in of combinational circuits, for $ISoTT$ the maximum implicant size can be used as K_{max}. In this sense, $ISoTT$ can be regarded as a DNF learning algorithm [3].

3.3 AES Accelerator

$ISoTT$ is superior with regular and structured circuits: the more structure (or less entropy) is present in the circuit, the better $ISoTT$ can exploit it. $JSoTT$, however, is better suited to high entropy circuits characterized by the avalanche effect, which facilitates exploration of dependencies. We took the tiny AES core from the Open Cores repository [11] as an ultimate example of a high-entropy circuit. This is a highly pipelined implementation of the AES encryption/decryption, which contains 6848 internal registers. The maximum transitive fan-in of the circuit is 8. Nevertheless, thanks to the avalanche effect, K_{max} as low as 4 suffices for detecting all the dependencies. Hence, $JSoTT$ learns the AES circuit precisely and with little effort, despite its formidable size; see Table 1. The accuracy indicates the number of correctly reconstructed flip-flops. Because the algorithm is random, results differ slightly between invocations.

Table 1: AES Circuit Reconstruction

Selected K_{max}	Accuracy	Number of probes
2	44%	389109
3	93%	1251169
4	100%	1594819

Note that cryptographic functions, considered complex, are actually an easy target for our reverse engineering attack. Additionally, thanks to the scalability of $JSoTT$, the size of the circuit is less important than its structure. For example, a deeper pipeline increases the chance of successful reverse engineering thanks to the finer partitioning of the logic.

4. RELATED WORK

Several papers discuss non-invasive methods for reverse engineering. Kash et al. [12] propose an optical method for monitoring switching activity of different locations in the circuit. This method requires prior knowledge of the circuit and access to test patterns. The Side Channel Analysis for Reverse Engineering (SCARE) [20] employs power analysis to obtain design information. This information is coarse due to the natural limitations of the power analysis. [13] and [15] present various techniques addressing phase two of the reverse engineering process, behavioral model extraction. Our work focuses on the first phase, netlist extraction.

Saab et al. [17] first proposed using the scan to extract the functionality of an integrated circuit. They present a randomized algorithm that finds dependencies between registers and evaluate it with small benchmarks and an AES function. We add the discovery phase and provide a systematic learning method along with a discussion of the algorithms' complexity and scalability metrics. We also propose a heuristic algorithm tuned for certain types of (e.g. arithmetic) circuits. In [8], scan based techniques are combined with invasive methods to discover camouflaged cells. The authors use SAT solvers to discover the function of the camouflaged cells, achieving notable efficiency. Our non-invasive scan based attack assumes the entire logic cone as a black box.

Other papers propose countermeasures against the scan based side-channel attack. Da Rolt et al. [5] present their comprehensive classification, which we use to show that some of the countermeasures are ineffective against the scan based reverse engineering attack. For example, Da Rolt shows that the Advanced DFT Structure is insecure against the scan side channel attack in general. The BIST structure resists any scan based attack by disabling external access to the scan chains, assuming no bypass modes. This complicates debug and field failure diagnostics. In addition, fault coverage of BIST is insufficient. Hence, pure BIST solutions are rarely selected by vendors. The scrambling and modified scan chain types present an additional countermeasure, but they are inefficient against the reverse engineering attack. The attack is still possible in the presence of the aforementioned structures, but the output of the reverse engineering will include both the functional and the protection circuits. The adversary can then distinguish between the two. Finally, the Secret-free Test solution prevents volatile data from being exposed but does not prevent reverse engineering of the design data. Solutions of the types Access Restriction and Secure Wrapper prevent unauthorized access to scan chains. Thus, while they are also efficient against reverse engineering, combined attacks are possible that first target the authorization mechanism, for example using DPA [18].

Previous work relies on the circuit's ability to switch between mission and scan modes. One of the popular countermeasures against the scan-based attacks is thus to enforce reset when switching between the modes [9]. Because the reverse engineering attack retrieves static design information from the scan chains, it is immune to this countermeasure.

5. CONCLUSIONS AND FUTURE WORK

The reverse engineering attack is a threat for devices that rely on the secrecy of their proprietary algorithms and protection mechanisms. In this paper, we demonstrated a new attack that allows for the non-invasive reverse engineering of an entire device at the logical netlist level. We introduced new methods that exploit the limited transitive fan-in property for efficient learning. Using an algorithm that employs junta learning, we successfully reconstructed a hardware implementation of the AES cryptographic function. Hence, our experimental results show that exploitation of the scan side channel for reverse engineering is a real threat. It can give the adversary full or partial information about the circuit contents. Even partial information may reveal sufficient data to serve malicious purposes. The presented attack is also immune to some popular countermeasures.

Our future work will focus on examining algorithms that are more efficient or are tuned for specific circuit types. *JSoTT*'s scalability makes the presented method potentially applicable also to large scale SoC devices. Combining *JSoTT* with algorithms that exploit additional heuristics (such as *ISoTT*) can target complex logic within large devices. Of special interest is when some part of the circuit is known a priori. Here, the reverse engineering technique may help in detecting hardware Trojans.

6. ACKNOWLEDGMENTS

This research was partially supported by the Hasso Plattner Institute (HPI).

7. REFERENCES

[1] L. Azriel, R. Ginosar, S. Gueron, and A. Mendelson. Using Scan Side Channel for Detecting IP Theft. In *HASP 2016*, pages 1–8, NY, USA, 2016. ACM Press.

[2] L. Azriel, R. Ginosar, and A. Mendelson. Exploiting the Scan Side Channel for Reverse Engineering of a VLSI Device. Technical Report CCIT Report 897, Technion, Israel Institute of Technology, 2016.

[3] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. *Journal of Computer and System Sciences*, 68(1):205–234, Feb 2004.

[4] F. Corno, M. S. Reorda, and G. Squillero. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Computers*, 17(3):44–53, 2000.

[5] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. Test Versus Security: Past and Present. *IEEE Transactions on Emerging Topics in Computing*, 2(1):50–62, Mar 2014.

[6] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. New security threats against chips containing scan chain structures. *IEEE HOST 2011*, pages 110–110, Jun 2011.

[7] P. Damaschke. Adaptive Versus Nonadaptive Attribute-Efficient Learning. *Machine Learning*, 41(2):197–215, 2000.

[8] M. El Massad, S. Garg, and M. V. Tripunitara. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes. In *NDSS*, 2015.

[9] D. Hely, F. Bancel, M. Flottes, and B. Rouzeyre. Test Control for Secure Scan Designs. *European Test Symposium (ETS'05)*, pages 190–195, 2005.

[10] D. Hely, K. Rosenfeld, and R. Karri. Security challenges during VLSI test. In *IEEE 9th International New Circuits and systems conference*, pages 486–489. Ieee, Jun 2011.

[11] H. Hsing. tiny_aes IP project, 2012.

[12] J. Kash, J. Tsang, and D. Knebel. Method and apparatus for reverse engineering integrated circuits by monitoring optical emission, Dec 2002.

[13] W. Li, Z. Wasson, and S. A. Seshia. Reverse Engineering Circuits Using Behavioral Pattern Mining. In *IEEE HOST*, pages 83–88, 2012.

[14] O. B. Lupanov. On circuits of functional elements with delay. *Probl. Kibern*, (23):43–81, 1970.

[15] K. S. McElvain. Methods and apparatuses for automatic extraction of finite state machines, 2001.

[16] R. O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.

[17] D. G. Saab, V. Nagubadi, F. Kocan, and J. Abraham. Extraction based verification method for off the shelf integrated circuits. In *1st Asia Symp. on Quality Electronic Design*, pages 396–400. IEEE, Jul 2009.

[18] S. Skorobogatov and C. Woods. Breakthrough silicon scanning discovers backdoor in military chip. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428, pages 23–40. Springer, Sep 2012.

[19] R. Torrance and D. James. The State-of-the-Art in IC Reverse Engineering. In C. Clavier and K. Gaj, editors, *CHES 2009*, volume 5747, pages 363–381. Springer Berlin Heidelberg, Aug 2009.

[20] X. Wang, S. Narasimhan, A. Krishna, and S. Bhunia. SCARE: Side-Channel Analysis Based Reverse Engineering for Post-Silicon Validation. In *25th Intl. Conf. VLSI Design*, pages 304–309. IEEE, Jan 2012.