Hardware Trojan Detection Using Exhaustive Testing of k-bit Subspaces

Nicole Lesperance, Shrikant Kulkarni, and Kwang-Ting (Tim) Cheng
ECE Department - UC Santa Barbara
Santa Barbara, CA 93106, USA
{nlesperance, skulkarni, timcheng}@ece.ucsb.edu

Abstract— Post-silicon hardware Trojan detection is challenging because the attacker only needs to implement one of many possible design modifications, while the verification effort must guarantee the absence of all imaginable malicious circuitry. Existing test generation strategies for Trojan detection use controllability and observability metrics to limit the modifications targeted. However, for cryptographic hardware, the n plaintext bits are ideal for an attacker to use in Trojan triggering because the size of n prohibits exhaustive testing, and all n bits have identical controllability, making it impossible to bias testing using existing methods. Our detection method addresses this difficult case by observing that an attacker can realistically only afford to use a small subset, k, of all n possible signals for triggering. By aiming to exhaustively cover all possible k subsets of signals, we guarantee detection of Trojans using less than k plaintext bits in the trigger. We provide suggestions on how to determine k, and validate our approach using an AES design.

I. Introduction

Hardware Trojans are a major concern for both semiconductor design houses and the U.S. government [1]. Economic factors mandate that design, manufacturing, testing, and deployment of silicon chips is a global effort involving multiple companies and countries. If a single contributor in this process decides it advantageous to insert malicious functionality into the chip, referred to as *Hardware Trojans*, the consequences can be disastrous.

Goals of Hardware Trojans range from denial of service attacks such as premature aging and bus deadlock to subtler attacks which attempt to gain undetected privileged access on a system, leak secret information through side channels, or weaken random number generator output [2].

A hardware Trojan consists of two components: a **trigger** and **payload**. The trigger is a set of conditions that, when met, cause a Trojan to exercise its malicious functionality. These conditions range from none, such as in [3], which continuously leaks AES secret key bits by subtly modifying the circuit's power consumption, to the occurrence of very specific patterns or sequences of analog or digital signals, such as in [4], where a Trojan inserted in an AES circuit waits for a sequence of 3 specific plaintext values before delivering its payload.

The payload realizes the goal of the Trojan circuitry, and can either affect circuit functionally or leak information without causing "incorrect" circuit behavior. **Successful detection of Trojan circuitry must *both* activate the Trojan and propagate evidence of malicious behavior to observable points.**

Existing techniques used to detect hardware Trojans in a large chip population fall into two main categories:

1. Techniques identifying anomalies in chip side-channel characteristics such as power consumption and delay
2. Techniques identifying Trojans in the functional domain by improving circuit observability and activation likelihood

Techniques which identify chips containing Trojans by comparing side-channel characteristics such as power [5, 6] and delay [7] with Trojan-free chips or models derived from the Trojan-free netlist, unlike functional testing, have the ability to identify malicious behavior which does not affect any values of *known* circuit nodes.

However, side-channel detection methods face ever-increasing process variation, which can overshadow the influence a Trojan has on the chip signature, especially for large complex designs such as SoCs. Beyond this, these focus mainly on the detection mechanism, and still rely on the ability of the test vectors or placement of scan flip-flops to partially or fully activate the Trojan.

Determining which design states should be explored during testing to activate Trojan circuitry, and how to best propagate Trojan behavior to observable points is an important task, on which the effectiveness of both side-channel and functional detection methods rely on.

Our work presents a post-silicon test vector generation strategy, especially applicable to cryptographic hardware, that detects Trojans with triggers based on patterns and sequences of digital signals. A stealthy Trojan has a very small probability of being activated during both the verification effort and during normal operation, but is relatively easy for the adversary to force.

Many existing methods for post-silicon test vector generation, such as [8, 9, 10, 11], use node controllability and observability to bias the test set. The assumption is that in order to decrease Trojan activation probability, the adversary will select signals that have very low 0 or 1-controllability, making the combination of these rare values unlikely to occur during testing. These strategies first identify random-pattern resistant nodes in the circuit, and their corresponding rare values, then derive

an optimal test set to trigger low probability node values multiple (N) times.

The usability of a Trojan from the attacker's point of view is severely diminished if the attacker cannot reliably control the triggering signals. Existing methods assume all inputs are attacker controllable, hence every single node is a candidate triggering signal for an attacker with complete knowledge of the design.

In cryptographic hardware, the key bits are unknown to the attacker, therefore any internal circuit nodes influenced by key bits will be uncontrollable, hence cannot be reliably used as triggering signals. For example, in AES, the first step is to XOR the plaintext with key bits [12], leaving the plaintext bits as the only viable triggering signals. For many Trojans proposed for AES [4, 13] and RSA [6], this is indeed the case, and testing strategies based on rare circuit values are not applicable since all plaintext bits have identical controllability and observability.

Challenge-response protocols implemented on both servers and embedded systems such as smart cards, allow the challenger to select the plaintext. In the case of AES, this gives the attacker 128 bits to choose from.

Figure 1 illustrates the AES Trojan implemented in [4] and [13]. Both works use a subset, k, of n bits (where $n = 128$), and the Trojan payload implements Piret's differential fault attack [14], allowing recovery of all secret key bits after observing as few as 2 faulty ciphertexts.

Even if the attacker utilizes only a small subset of the 128-bit plaintext, unless the verification team can discover which subset the attacker will use, they are left facing the impossible task of verifying all 2^{128} plaintext values.

Since exhaustive testing is infeasible, our solution makes the following observation: **An attacker can realistically only afford to use a small subset, k, of all n possible controllable signals for triggering.**

Our Trojan detection strategy uses this observation, instead of controllability and observability metrics, to reduce the state space targeted by the test vectors. To our knowledge, this is the first work to address scenarios where controllability and observability metrics do not provide a foothold for biasing testing. Our approach is exhaustive, but with respect to k instead of n, meaning that our test vectors are *guaranteed* to activate a Trojan if the k-value chosen is realistic.

Section IV.B discusses the factors involved in determining k, but intuitively a hardware Trojan containing a 128-bit comparator or large counter will have a noticeable area and power footprint. The feasibility of inserting such circuits is far less during fabrication, but Trojans inserted pre-silicon have the opportunity to be detected by formal methods, simulation, and analysis of the RTL code. Our work also provides additional strategies for the case where one cannot afford exhaustive testing with respect to the estimated k-value.

The rest of the paper is organized as follows: Section II specifies the class of Trojans our solution detects and relates their activation to the concept of k-subspace coverage, Section III details how to generate test vectors providing exhaustive k-subspace coverage, Section IV presents a case study where different Trojan triggers are inserted in a 128-bit AES circuit and discuss how area overhead metrics can influence the selection of k, and Section V summarizes our contributions.

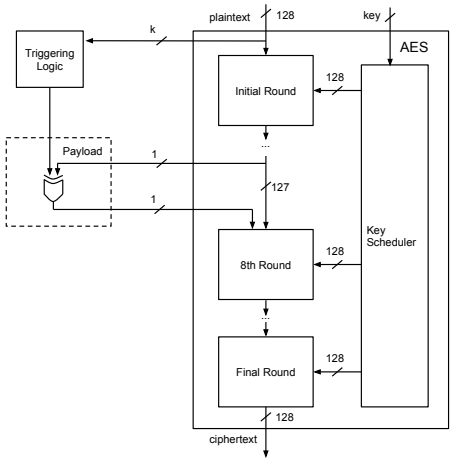

Fig. 1. AES Fault Attack Trojan

II. Problem Definition and Formulation

A. Interaction with Existing Test and Detection Methods

Traditional manufacturing tests target stuck-at or delay fault models, based on circuit structure. Trojans inserted during or after fabrication are not present in the gate-level model therefore are not targeted by test pattern generation tools or candidates for observation points.

Because of the confusion and diffusion properties of cryptographic algorithms, the difficultly in Trojan detection lies in triggering the Trojan, not propagating faulty behavior cause by the Trojan to observable points. For example, a Trojan payload may shorten the number of rounds in a cipher or create a fault during encryption. In both cases, the resulting cipher text will differ from the Trojan free version, and is easily detectable.

Therefore, for the remainder of the paper we focus on test generation strategies for trigger activation, and do not address the class of Trojans which leak circuit information through side channels. However, our method can be used in conjunction with existing side channel detection methodologies to magnify the difference between Trojan and Trojan-free side channel fingerprints in the case where information leakage only occurs after a triggering condition is met.

B. Trojan Trigger Models

Our work aims to detect Trojans whose digital triggers take as input k design signals, where $0 < k \leq n$, and n is the total number of attacker controllable signals.

For the attacker, increasing k *decreases* the probability that the Trojan is triggered during testing, however an increase in k leads to a larger Trojan area and power footprint, making the circuitry more visible.

We consider 3 classes of k-bit triggers:

TABLE I
ACTIVATION PROBABILITIES DURING A SEQUENCE OF t n-BIT UNIFORM RANDOM TEST VECTORS FOR EACH TRIGGER MODEL

Combinational	**Partial Ordering**
$1-\left(1-\frac{1}{2^k}\right)^t$	$1-\left(1-\frac{1}{2^{km}}\right)^{\binom{t}{m}}$
Contiguous Ordering	
$1-\left(1-\frac{1}{2^{km}}\right)^{t-m+1}$	

1. **Combinational:** activation occurs immediately upon recognition of a k-bit pattern

2. **Partially Ordered Sequence:** activation occurs upon recognition of a partially ordered sequence of m k-bit patterns

3. **Contiguously Ordered Sequence:** activation occurs upon recognition of a contiguous sequence of m k-bit patterns

The Trojan activation probabilities during a sequence of t n-bit uniform random test vectors for trigger classes 1 - 3 are given by the equations in Table I. Depending on the desired Trojan area overhead and activation probability, the attacker can implement any of the 3 trigger types inside the Triggering Logic block in Figure 1.

It should be noted that for the partial and contiguous orderings, the m patterns need not be unique. However, implementing even a few different k-bit pattern recognizers leads to an significant increase in Trojan area, as seen in Table VII in Section IV.A, without decreasing activation probability. Therefore, it is very reasonable to assume that only 1 k-bit pattern is used in conjunction with a counter.

Counting m patterns before triggering greatly reduces activation probability. A special case of trigger classes 2 and 3 is a large counter that counts clock cycles instead of patterns. [15] refers to this type of trigger as a "time bomb", and proposes periodically performing power resets during circuit operation to limit the maximum counter value, forcing the attacker to use a smaller counter if the Trojan is ever to trigger in the field. If circuit validation is run for a time period exceeding the power reset period, the Trojan is guaranteed to be triggered. For the more general class of sequential triggers that we are considering, power resets effectively limit the value of m, but the test vectors must still ensure the appearance of the magic k-bit pattern m times before activation.

C. Subspace Coverage

Let n be the number of possible attacker-influenced circuit nodes. Some examples are the plaintext bits in cryptographic hardware or bus data bits on a processor running untrusted software or firmware.

If a Trojan can incorporate a maximum of k bits into its triggering mechanism, the goal of the detection effort is to apply the smallest number of n-bit test vectors, $|T_{min}(n,k)|$, which exhaustively cover all 2^k possible values that can occur on all $\binom{n}{k}$ possible sets of k-bit signals (k-subspaces), guaranteeing Trojan activation.

Figure 2 shows all 6 possible 2-subspaces when $n = 4$. One simple method of generating the test vectors for this set is to target each subspace individually, resulting in $\binom{n}{k}\times 2^k = \binom{4}{2}\times 2^2 = 24$ test vectors. However, since only 16 test vectors are needed to exhaustively test 4-bits, it is obvious that this method does not generate $T_{min}(n,k)$.

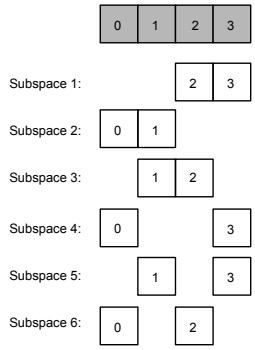

Fig. 2. All 6 2-bit subspaces in a 4-bit vector

An example exhaustive 2-subspace test set generated by trial and error contains only 5 vectors: $\{0000, 0111, 1110, 1101, 1011\}$. These 5 vectors guarantee activation of a trigger using *any* 2 out of 4 controllable bits matching *any* 2-bit pattern.

Clearly, $2^k \leq |T_{min}(n,k)| \leq 2^n$, but it is not obvious how to generate $T_{min}(n,k)$ systematically, or determine $|T_{min}(n,k)|$.

III. OUR SOLUTION

A. Test Generation for Exhaustive k-subspace Coverage

A method for generating several sets of n-bit test vectors which exhaustively cover all k-subspaces is given in [16]. Each test set is composed of 1 or more sets of *constant weight vectors*. A set of constant weight vectors is the set of *all* n-bit vectors with a given Hamming weight w. There are $\binom{n}{w}$ vectors in a constant weight set.

There are $n-k+1$ test sets to choose from, and each is described by a set of weights, which are found by solving Equation 1 with $n-k+1$ different values for c.

$$w \equiv c \bmod (n-k+1),\ 0 \leq c \leq n-k \tag{1}$$

The number of test vectors in each test set is

$$\sum_{all\ weights} \binom{n}{w_i} \tag{2}$$

For example, let $n = 8$ and $k = 3$. Equation 1 becomes

$$w \equiv c \bmod (6),\ 0 \leq c \leq 5 \tag{3}$$

There are 6 different test sets which can exhaustively cover all 3-subspaces. The weights and test lengths are given in the chart below.

Clearly, not all generated test sets are optimal. The test set composed of all vectors with Hamming weights 1 and 7 is the smallest. The weights for the smallest test set are given by Equation 4. The size of the minimal test set, $|T_{min}(n,k)|$, is given by Equation 5 [16].

$$w_0 = \left\lfloor \frac{k}{2} \right\rfloor,\ w_1 = \left\lfloor \frac{k}{2} \right\rfloor + (n-k+1) \tag{4}$$

TABLE II
TEST SETS FOR $n = 8$, $k = 3$

c	Weight Set	Test Length, $\lvert T(8,3)\rvert$
0	$\{0, 6\}$	29
1	$\{1, 7\}$	16
2	$\{2, 8\}$	29
3	$\{3\}$	56
4	$\{4\}$	70
5	$\{5\}$	56

$$|T_{min}(n,k)| = \binom{n}{\lfloor \frac{k}{2} \rfloor} + \binom{n}{k - \lfloor \frac{k}{2} \rfloor - 1} \quad (5)$$

B. Sequential Triggers

If the Trojan is triggered by partially or continuously ordered sequences of m k-bit patterns, the minimal k-subspace exhaustive test set, $T_{min}(n,k)$, provided in the previous section does not guarantee activation.

Partial Ordering: All *partially* ordered sequences of m k-bit patterns can be exhaustively tested using $|T_{min}(n,k)| \times m$ vectors by repeating $T_{min}(n,k)$ m times.

Contiguous Ordering: If the m patterns can be distinct, then $|T_{min}(n,k)|^m$ test vectors are needed to exhaustively cover this scenario! If the same k-bit pattern, occurring m times in a row, triggers the Trojan, we can repeat each vector in $T_{min}(n,k)$ m times to exhaustively cover this case using only $|T_{min}(n,k)| \times m$ test vectors.

C. Example Test Set Sizes

Table III illustrates how n, m, and k affect the test set size for exhaustive k-subspace testing. For Trojans with $m > 1$, Table III shows the test length assuming either partial ordering of m possibly distinct patterns, or contiguous ordering of m identical patterns.

TABLE III
TEST SET LENGTH FOR EXHAUSTIVE k-SUBSPACE COVERAGE

n	m	k	Test Set Length
128	1	2	2^7
128	1	4	2^{13}
128	1	8	2^{23}
128	1	16	2^{40}
128	1	32	2^{67}
128	4	8	2^{25}
128	8	8	2^{26}
128	10000	8	2^{37}
256	1	8	2^{27}
256	10000	8	2^{41}
2048	1	8	2^{39}
2048	10000	8	2^{53}

Increasing k and n results in exponential growth in test size. Increasing m causes linear growth in test size, and can be limited by using the power reset technique [15].

D. When Exhaustive k-subspace Testing is Too Expensive

Although exhaustive k-subspace coverage requires fewer than 2^n vectors, test size grows exponentially with increases in n and k, as seen in Table III, in some cases making exhaustive testing infeasible. Let k_{max} be the maximum number of bits a Trojan trigger can utilize, and T_{max} be the number of vectors budgeted for testing.

When $|T(n,k_{max})| \leq T_{max}$, the exhaustive k_{max}-subspace test set is both guaranteed to activate the Trojan and within T_{max}. However, when $|T(n,k_{max})| > T_{max}$, other testing strategies must be considered.

Since our method aims to detect Trojans in designs where signal controllability and observability cannot guide test vector selection, the only alternative testing strategy is the application of uniform random vectors to the attacker controllable bits in the design. Depending on T_{max}, n, k, and k_{max}, one can consider the following test sets:

- **Strategy 1:** T_{max} uniform random vectors
- **Strategy 2:** The complete exhaustive k-subspace test set where $k < k_{max}$, and $|T(n,k)| \leq T_{max}$
- **Strategy 3:** The complete exhaustive k-subspace test set where $|T(n,k)| \approx \frac{T_{max}}{2}$ in addition to $\frac{T_{max}}{2}$ random vectors (excluding those already in $T(n,k)$)

The Trojan activation probability, p_a, for Strategy 1 is given in Table I, p_a for Strategy 3 is computed using simulation, while the derivation of p_a for Strategy 2 is given below.

Strategy 2 p_a: The probability of observing a random k_{max}-bit pattern in the k-subspace exhaustive test set $T(n,k)$, where $k < k_{max}$, can be derived by considering the possible Hamming weights, w_{troj}, for the k_{max}-bit triggering pattern. $0 \leq w_{troj} \leq k_{max}$.

$T(n,k)$ contains *all* possible n-bit vectors with Hamming weights $\{w_0, w_1\}$ given by Equation 1. $T(n,k)$ is guaranteed to activate a Trojan with weight w_{troj} if the $k_{free} = n - k_{max}$ bits unused by the Trojan can be assigned a Hamming weight x such that

$$w_{troj} + x = w_0 \text{ or } w_1 \quad (6)$$

$$f(T, w_{troj}, k_{max}) = \begin{cases} 0 & \nexists \text{ a solution to Eq. 6} \\ 1 & \exists \text{ a solution to Eq. 6} \end{cases} \quad (7)$$

By enumerating all possible trigger pattern Hamming weights and considering how many such patterns exist for a given n, the number of patterns detectable by any given $T(n,k)$ can be computed, leading to the formula for activation probability given in Equation 8.

$$p_a = \frac{\sum_{w_{troj}=0}^{k_{max}} \binom{k_{max}}{w_{troj}} \times f(T, w_{troj}, k_{max})}{2^{k_{max}}} \quad (8)$$

Comparisons and Discussion: Table IV compares the activation probability given for Strategy 2 with the activation probability for T_{max} uniform random vectors when $n = 128$. Table V shows how the mixed test set

TABLE IV
ACTIVATION PROBABILITIES FOR $n = 128$

k_{max}	k	$\lvert T(n,k)\rvert$	p_a $(T(n,k))$	p_a (rand)
16	4	2^{13}	0.00235	0.1184
16	8	2^{23}	0.04904	0.9999
32	4	2^{13}	1.309e-07	1.922e-06
32	8	2^{23}	1.093e-05	0.00256
32	16	2^{40}	0.004551	0.9999
64	4	2^{13}	1.163e-16	4.476e-16
64	8	2^{23}	3.919e-14	5.968e-13
64	16	2^{40}	3.163e-10	8.263e-08

TABLE V
ACTIVATION PROBABILITIES FOR $n = 128$

k_{max}	k	$\lvert T(n,k)\rvert, \lvert T_{rnd}\rvert, \lvert T_{total}\rvert$	p_a (mixed)	p_a (rand)
8	3	$256, 256, 512$	0.6574	0.8652
16	3	$256, 256, 512$	0.0036	0.007782
20	3	$256, 256, 512$	0.0003	0.0004882
10	5	$2^{14}, 2^{14}, 2^{15}$	0.9999	0.9999
16	5	$2^{14}, 2^{14}, 2^{15}$	0.213	0.3911
20	5	$2^{14}, 2^{14}, 2^{15}$	0.017	0.03053

(Strategy 3), compares with the uniform random vectors for $n = 128$ and various values of k and k_{max}. $|T(n,k)|$ is the number of test vectors in the exhaustive k-subspace test set, $|T_{rnd}|$ is the number of weighted random vectors used, and $|T_{total}| = |T(n,k)| + |T_{rnd}|$.

Exhaustive k-subspace test sets always have lower activation probabilities than the same number of uniform random vectors when $k_{max} > k$. This is because uniform random vectors sample from the entire space of 2^n possible vectors while $T(n,k)$ is restricted to vectors of particular Hamming weights. As k_{max} becomes larger compared to k, there are more Trojan trigger Hamming weights not targeted by exhaustive k-subspace vectors that uniform random vectors still sample from.

The advantage of using exhaustive k-subspace test vectors for a feasible k is that **activation for all subspaces smaller than k is guaranteed.** Because hardware Trojan insertion is challenging, especially during fabrication, k values smaller than k_{max} are more likely, and exhaustively testing these possibilities is advantageous.

If random vectors can be used in combination (Strategy 3) to target the less likely larger k values, the activation probability is closer to that of the uniform random vectors. Strategy 3 provides a balance between guaranteed activation for smaller k and optimal sampling of the remaining state space with random vectors.

IV. AES TROJAN CASE STUDY

A. Area Overhead

We have implemented the Trojan in Figure 1 for each of the 3 Trojan trigger types shown in Section II.B in an AES encryption IP from OpenCores [17], where $n = 128$.

The infected designs were synthesized in 45nm technology using the NanGate Open Cell Library [18] with Synopsys Design Compiler (ver I-2013.12-SP2) and routed using Cadence Encounter (v09.14) to quantify the overhead due to the trojan logic. The percentage increase in area for the infected design and equivalent 2-input NAND gate count of the Trojan is shown in Table VI for various m and k values.

TABLE VI
$n = 128$, TROJAN % AREA INCREASE, **G:** GATE COUNT (EQUIVALENT 2-INPUT NAND GATES) (m IDENTICAL PATTERNS)

	m							
	1		128		1024		8192	
k	%	G	%	G	%	G	%	G
4	0.11	25	0.77	176	1.00	230	1.23	282
8	0.14	32	0.83	190	1.06	243	1.29	295
32	0.32	72	1.18	270	1.41	323	1.64	376
64	0.55	125	1.65	377	1.88	430	2.11	482
128	1.01	232	2.58	590	2.82	644	3.04	695

In Table VI, the m patterns are identical, not distinct. It can be seen in Table VII that if the trigger is designed with multiple distinct patterns, the area overhead increases significantly. For example, when $k = 4$ and the number of distinct patterns is 4, the area overhead is already greater than 1% of the original design.

The synthesis area and NAND gate-count increase significantly as k and m increase, more sharply with an increase in k than that of m. This is because the value of m doubles with addition of only a single counter bit, while increasing k requires an increase in the comparison logic of the trigger. Thankfully, a limit on m can be enforced by using the power reset strategy outlined in [15].

TABLE VII
$n = 128$, TROJAN % AREA INCREASE, **G:** GATE COUNT (EQUIVALENT 2-INPUT NAND GATES) ($m = 8192$, MULTIPLE DISTINCT PATTERNS)

	# distinct patterns					
	4		8		16	
k	%	G	%	G	%	G
4	1.36	310	1.82	417	2.72	621
8	1.48	338	2.05	470	3.18	728
32	2.17	497	3.46	790	5.98	1368

B. Factors Influencing k_{max}

Determining the feasibility of different k_{max} values requires formulating a realistic threat model for each design and testing scenario. Our method can target Trojans inserted both pre-silicon and during fabrication, but the ease of Trojan insertion and the variety of detection methodologies available at both stages differs greatly.

Trojan Insertion Pre-silicon: On one hand, Trojans inserted in 3rd party IP have practically no limits

on Trojan size, since the customer often only has access to the net list or a pre-routed block, making it difficult to reverse engineer the design and identify Trojans or detect increases in area due to Trojan circuitry. Also, post-silicon side-channel detection methods will fail due to the lack of Trojan-free gate-level models, as well as the lack of golden reference chips.

On the other hand, complete observability during simulation and the availability of formal methods such as equivalence checking provide powerful opportunities for detection strategies such as [19]. If a Trojan is inserted at gate level (post-synthesis), and attempts to hide within minor changes made during the effort to meet timing and power requirements, the presence of several hundred extra gates will surely be noticed, as is the case for when $k > 32$, and $m > 128$ as seen in Table VI. How reverse engineering, code and circuit analysis, and formal methods can be used to either prove $k_{max} = 0$ or determine a reasonable k_{max} to target using our post-silicon exhaustive k-subspace approach is a topic for further research.

Trojan Insertion During Fabrication: Modifying the optical mask to insert Trojans is extremely difficult. Only a few works have actually fabricated circuits containing hardware Trojans and analyzed the complexity of insertion at mask level. In [13], Trojans instrumenting Piret's fault attack on an AES circuit are inserted into the layout using a commercial Engineering Change Order (ECO) placement tool. They vary the number of plaintext bits used in the trigger until the software is no longer able to place the ECO without completely re-routing the entire design.

With a Core Utilization Rate of 99%, the tool cannot place an ECO for a Trojan composed of as few as 16 AND gates. While further research is required to validate this approach for estimating an upper bound on k_{max}, it is clear that k is severely restricted for mask level Trojan insertion, making exhaustive k-subspace testing a feasible and complete method for guaranteeing Trojan activation.

V. Summary and Conclusions

Our AES circuit case study shows that realistically, an attacker can only incorporate k out of all n possible controllable signals into a Trojan triggering mechanism, where $k << n$. We use this observation instead of the controllability and observability metrics widely employed in existing methods to guarantee detection of Trojans in cryptographic circuits using up to k triggering signals. We also present additional strategies when the size of k requires a prohibitively large exhaustive test set to guarantee detection.

Acknowledgements

We would like to acknowledge Professor Çetin Kaya Koç for his valuable insight regarding this work.

References

[1] S. Adee, "The hunt for the kill switch," *IEEE Spectr.*, vol. 45, pp. 34–39, May 2008.

[2] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Des. Test*, vol. 27, pp. 10–25, Jan. 2010.

[3] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels.," in *ICCAD*, pp. 117–122, IEEE, 2009.

[4] S. Ali *et al.*, "Multi-level attacks: An emerging security concern for cryptographic hardware," in *DATE, 2011*, pp. 1–4, March 2011.

[5] Y. Liu, K. Huang, and Y. Makris, "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," DAC, 2014.

[6] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using ic fingerprinting," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pp. 296–310, May 2007.

[7] K. Xiao, X. Zhang, and M. Tehranipoor, "A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay," *IEEE Design & Test of Computers*, vol. 30, no. 2, pp. 26–34, 2013.

[8] F. Wolff, C. Papachristou, S. Bhunia, and R. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *DATE '08*, pp. 1362–1365, March 2008.

[9] R. S. Chakraborty *et al.*, "Mero: A statistical approach for hardware trojan detection," in *CHES 2009*, vol. 5747 of *LNCS*, pp. 396–410, Springer Berlin Heidelberg, 2009.

[10] S. Narasimhan *et al.*, "Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach," in *HOST'10*, June 2010.

[11] A. Sreedhar, S. Kundu, and I. Koren, "On reliability trojan injection and detection," *J. Low Power Electronics*, vol. 8, no. 5, pp. 674–683, 2012.

[12] http://csrc.nist.gov/publications/fips/fips197/fips 197.pdf, "Advanced encryption standard (aes)."

[13] S. Bhasin *et al.*, "Hardware Trojan Horses in Cryptographic IP Cores," FDTC '13, (Washington, DC, USA), pp. 15–29, IEEE Computer Society, 2013.

[14] G. Piret and J.-J. Quisquater, "A differential fault attack technique against spn structures, with application to the aes and khazad," in *CHES, Sep. 2003*, vol. 2779 of *LNCS*, pp. 77–88, Springer, 2003.

[15] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," in *Security and Privacy (SP), 2011 IEEE Symposium on*, pp. 49–63, May 2011.

[16] D. Tang and L. S. Woo, "Exhaustive test pattern generation with constant weight vectors," *Computers, IEEE Transactions on*, Dec 1983.

[17] "Aes (rijndael) ip core," 2002. http://opencores.org/project,aes_core.

[18] "Nangate 45nm open cell library," 2011. https://www.si2.org/openeda.si2.org/projects/nangatelib.

[19] M. Banga and M. Hsiao, "Trusted rtl: Trojan detection methodology in pre-silicon designs," in *HOST'10*, pp. 56–59, June 2010.