Journal of Hardware and Systems Security manuscript No.
(will be inserted by the editor)

On the Limitations of Obfuscating Redundant Circuits in Frustrating Hardware Trojan Implantation

Nathanael Weidler · Ryan Gerdes · Thidapat Chantem

Received: date / Accepted: date

Abstract Split manufacturing is a method to secure circuits by creating layers of a circuit separately—one layer is manufactured at a trusted foundry and the other at an untrusted foundry. The complete circuit is unknowable without both pieces, thus the circuit can't be effectively manipulated by, e.g., inserting a hardware Trojan at manufacture time. A prominent example of this approach is the work "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation" [11]. In the work it is claimed that even if an attacker knows the exact layout of a circuit before division, the technique set forth would prevent the attacker from inserting an efficient (i.e., undetectable) hardware Trojan into the circuit unless they possessed knowledge of the trusted layer. This paper is notable because it gives strong theoretical reasons, as opposed to only providing empirical results, to suggest that the proposed method provides security for circuits.

In this work we examine whether this particular split manufacturing approach is effective in protecting redundant circuits, such as implementations of cryptographic ciphers, from the implantation of hardware Trojans. We show that it is indeed possible to insert a Trojan with a much higher success rate, and smaller footprint, than the example discussed in [11], which implies that, at least for this class of circuits, obfuscation provides significantly less security than the authors' theoretical analysis would suggest. To demonstrate its general applicability, our analysis is carried out not only on a the same type of circuit used as an example in [11] (an implementation of the Data Encryption Standard (DES)) but also an Advanced Encryption Standard (AES) circuit. For both circuits we demonstrate vast improvement for attacker success using the metrics used in [11].

N. Weidler
Utah State University
E-mail: nweidler@gmail.com
ORCiD: 0000-0003-4947-0460

R. Gerdes Virginia Tech E-mail: rgerdes@vt.edu
ORCiD: 0000-0003-0876-1181 · T. Chantem Virginia Tech E-mail: tchantem@vt.edu
ORCiD: 0000-0002-5688-5720

1 Introduction

Hardware Trojans have been widely discussed and analyzed [12,8,23,33,19,16]. Trojans in the hardware context are malicious modifications made to a circuit that can, for example, leak confidential information, such as secret keys, or cause a circuit to malfunction [23]. A hardware Trojan must be inserted at some point during the development of an integrated circuit (IC). There are four possible points of insertion during this cycle: the specification phase, the design phase, the fabrication phase and the assembly phase [30]. Hardware Trojans present a potential security concern to anything system or device that utilizes integrated circuits. The U.S. Department of Commerce has estimated that counterfeit electronic components have appeared in 39% of the Department of Defense supply chain [25]. Such counterfeit parts have been discovered in Navy helicopters and Air Force planes [4]. Thus, there exists a very real possibility that ICs that include malicious logic designed to leak critical information or make systems fail could make their way into either military or civilian systems.

In their efforts to thwart malicious actors, researchers have discovered new forms of Trojans and ways to defeat them [12]. Others have concentrated on identifying ways to detect hardware Trojans in a compromised circuit [20,21]. Another approach taken to mitigate these threats is to harden circuits against hardware Trojan implantation. This approach creates circuits in such a way that the difficulty of successfully inserting a Trojan is significantly increased, if not impossible. An example of this approach was proposed by Imeson et al. [11], whereby a wire lifting procedure, coupled with split fabrication, was used to obscure the target circuit from the attacker. This method concentrated on obfuscating the circuit: if an attacker cannot distinguish different parts of the circuit from one another, they would not be able to insert an effective hardware Trojan. This procedure provided a novel method to secure a circuit against hardware Trojan implantation at the foundry.

What distinguishes [11], and makes it an important work in the field of circuit obfuscation, is that it provides strong theoretical proofs for its security [31], unlike most of the literature on the topic, including more recent papers, which merely provide empirical results [31,18,28,29,27,32]. Given its status as a foundational work in the field, it is important to understand the limitations of its methods, which will be the focus of the current work.

Specifically, we show that the claim of increased security for circuits without regard to type is too broad (it is implicit in the work that the approach would have applicability to many types of circuits). While we do not dispute the claims of increased security for circuits generally, we do hypothesize that there exists a class of circuits for which the proposed obfuscation method does not provide the claimed security; viz., redundant circuits (defined in Section 3.3) with cryptographic circuits belonging to this this subclass of circuits.

In what follows we present methods to bypass the defense of [11] that we believe to be applicable not only the specific circuit examined therein (an implementation of the Data Encryption Standard (DES)) but also other redundant, pipelined circuits that are widely used in cryptography; e.g., the Feistel structure generally and substitution-permutation networks (SPN) [17,15,7,1,22]. The implementation of ciphers based on these structures are known to be vulnerable to fault injection attacks [13,2], to which our attack is a Trojan-based variant.

1.1 Contributions

A significant amount of time was spent in unsuccessfully trying to reproduce the results of the wire lifting procedure detailed in [11], including running the author's publicly available code on their example circuits. With this limitation in mind (i.e., that the original findings of [11] could not be replicated), we make the most pessimistic assumptions for an attacker in our work; i.e., we consider the most secure possible outcome for the result of the wire lifting procedure. Despite this we are able to show that the wire lifting procedure does not provide the intended amount of security for highly redundant circuits, particularly cryptographic circuits, e.g., implementations of DES and the Advanced Encryption Standard (AES). These algorithms were chosen to show that the wire lifting procedure does not frustrate the implantation of a hardware Trojan for either a Feistel or SPN structure.

To defeat the wire lifting procedure for DES we attack all portions of the circuit that are indistinguishable from one another at the same time, instead of choosing one portion and only attacking it, or attacking each portion one at a time, as is suggested in [11]. In attacking an AES circuit we do not attack every indistinguishable portion of the circuit at the same time but instead attack enough of the indistinguishable portions of the circuit to be able to recover the key through an exhaustive search. This allows the size of the Trojan to be less than it would have to be if every portion of the circuit indistinguishable from another were to be attacked. This method can also be applied to a DES circuit, increasing the probability of success of against a DES circuit to 100%.

1.2 Paper Organization

The motivation for our work and high-level findings are discussed in Section 2. Background on split manufacturing circuit obfuscation, the fault-style attack we consider to defeat obfuscation, and the types of circuits it is applicable to (redundant ones) and why they are vulnerable are detailed in Section 3. Section 4 discusses the threat model and details several weaknesses with split manufacturing obfuscation, specifically pertaining to cryptographic circuits based on Feistel structures, e.g., DES. Section 5 demonstrates how these weaknesses extend to other, non-Feistal structures, namely substitution–permutation networks (SPN), as exemplified by AES. Finally conclusions and future work are put forth in Section 6.

2 Motivation and Findings

With a work as highly regarded and influential as [11], it is important to point out limitations, without attempting to discredit the work as a whole, so that the method proposed is not erroneously applied to circuits for which the promise of increased security cannot be fulfilled.

For the circuit example (DES) given in [11] the claim is made that an attacker could either place a small Trojan in the circuit with a 1/256 chance of success or place a large, 1280 gate Trojan in the circuit with a 100% chance of success, though the number of plaintexts would increase by 255×. We present another possibility: a hardware Trojan containing only 256 gates. Simulations show a 75% chance of recovering the secret key with only two plaintexts presented using this Trojan. This is a 191× improvement over the success rate of the small Trojan presented, and at the same time, the Trojan comprises 5× fewer gates and requires exactly as many plaintexts as the small Trojan, not the 255× number required by the large Trojan.

We believe that the example of a cryptographic circuit was a poor choice for the wire lifting procedure not only because its size makes it time prohibitive to apply the method to (as discussed in Section 4), but also because these circuits are highly redundant in nature, which allows for effective fault-style attacks. DES has 16 identical rounds making it fit into the class of highly redundant circuits. It has been broken using differential fault analysis on early, middle and late rounds [24]. That is, even if a specific bit cannot be targeted because of the protections the wire lifting procedure provides, the circuit could still be compromised by a hardware Trojan. So long as faults can be induced in the DES circuit, even if the location of the Trojan or the round in which it was implanted in is not certain, the secret key can still be discovered (as shown in Sections 4 and 5).

Furthermore, we attempted to generalize the attack by investigating another highly redundant cryptographic circuit that implements AES, which uses a different cryptographic structure (SPN) from DES. We show that for an AES circuit the wire lifting procedure again does not provide the claimed security as the secret key can be recovered using at most six plaintexts (Section 5). Specifically, we show that with a Trojan of 640 gates the key can be recovered 53% of the time, 700 gates 89% of the time and an 800 gate Trojan would be able to recover the key 99.9% of the time. These Trojans are still 50%, 45% and 37.5% smaller than the style of Trojan suggested for DES (and adapted to AES), and they use 97.6% fewer plaintexts in the worst case.

We also feel it important to point out the difficulties we experienced attempting to reproduce the results of the DES example given in [11]. We were unable to perform the wire lifting procedure as outlined in the original work. We used the code base and examples from [11] that were made available to the public [10]. We have had a varying degree of success using this code. Over the course of several years, we have been in contact with the authors of the original work, who were at first very helpful in aiding our efforts to reproduce their work. They fixed errors in the code base and added files that were needed to build the code but were not originally included. The small example from the `README` provided does work, for instance. However, for the more complex circuits that are included with the code base, e.g., the DES circuit, the wire-lifting procedure never successfully terminated.

We note that one of the points in the original paper was that the wire lifting procedure is scalable, i.e., can be used on a large circuit like DES. We believe that our failure to produce a *k-secure* DES circuit, using the code provided by the authors, indicates a significant weakness of the original work. That is, although the wire lifting procedure, as implemented in the authors' code, works on smaller, simpler circuits, we have not been able to have a partitioned portion of the DES circuit complete the procedure. Even after gaining access to the cluster resources at the University of Utah and running the code for three weeks, the wire lifting procedure made little progress. We estimated that given the progress it would take additional years to complete. On another machine with the ability to run 24 threads and 128 GB of RAM, the example DES wire lifting procedure ran for over 135 days without finishing or showing significant progress. According to the authors, when queried, such computational resources should have been sufficient but they were unable to provide a time frame for completion.

3 Background

In this section, a brief background on circuit obfuscation is presented and followed by a short explanation of fault injection analysis. The term redundant circuits, as used in this work, is then defined.

3.1 Split Manufacturing Circuit Obfuscation

Imeson et al. introduced a wire lifting procedure to select wires to remove from the untrusted tier and place on the trusted tier. The wire lifting procedure is a greedy heuristic to make individual gates or groupings of gates indistinguishable from one another [11]. Without the traces between them on the untrusted tier the attacker cannot identify the intended location of his Trojan.

A gate is said to *k-secure* when there are $k-1$ other gates in the circuit that are indistinguishable from that gate. The higher number, the more secure the circuit. Imeson et al. define the *k-security* of the circuit as each

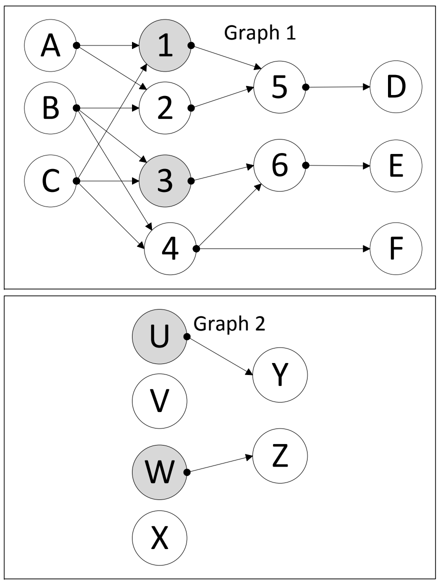

Fig. 1 Wire lifting example. Graph 2 has a *k-security* of 2, as each subgraph has at least one other that is identical to itself.

gate in the circuit being at least *k-secure*. As an example, after the wire-lifting procedure, if there were a set of gates that are 3-secure and the remaining gates are 5-secure, the *k-security* of the circuit would be 3-secure, as each gate is at least 3-secure. Depending on the target *k-security* of a circuit, more or fewer wires may need to be lifted.

Figure 1 shows an example circuit as an illustration of lifting wires to create *k-security*. Graph 1 represents a circuit in the original state with inputs `A`, `B`, and `C` and outputs `D`, `E`, and `F`. Graph 2 represents how the circuit would look after a wire lifting procedure to make the circuit 2-secure. Notice that the inputs and outputs are removed in Graph 2, as those wires have been entirely removed. In this example each gate, or subgraph, is indistinguishable from *2-1* or one other gate, or subgraph, making the circuit 2-secure. It is unknown if node `V` in Graph 2 represents node `2` or node `4` from Graph 1. Also, it is unknown if the subgraph of node `U` to node `Y` is the same as the subgraph of node `1` to node `5`, or node `3` to node `6`.

3.2 Fault Injection Analysis

The method of circuit obfuscation by wire-lifting set forth in [11] is said to protect a DES circuit against a hardware Trojan which would, essentially, create a fault attack on the least significant bit (LSB) of the fourteenth round. This technique of discovering the secret key by fault injection on the output of the fourteenth round or input of the fifteenth round is set forth in [24]. The technique is as follows: a known fault on the output of the fourteenth round propagates through the fifteenth and 16th rounds. (In this case, the known fault ensures a particular bit is always be zero.) By comparing a previously captured ciphertext which did not have the fault injected during it's encryption, to the ciphertext (using the same plain text) corrupted by the fault, the attacker is able to determine certain bits of the round key for the 16th round. The remaining bits of the round key are guessed and reversed through the DES key schedule to attain a DES secret key with eight missing bits. These final bits are then searched in a brute force manner by running the DES algorithm with each key and the plaintext that should give the uncorrupted cipher text. This process is iterated until a key is found that yield a match between the plaintext and the uncorrupted cipher text, at which point the entire secret key is known.

3.3 Redundant Circuits and *k-security*

We define a redundant circuit to be a circuit in which logic is duplicated multiple times, separated by pipeline stages. DES is a redundant circuit because it contains 16 identical rounds. AES is a redundant circuit because it contains nine identical rounds and a tenth round that is nearly identical to the first nine. The methods set forth in [11] are very good at creating portions of circuits that are indistinguishable from one another; however, the impact to the security of the method has not been explicitly considered in the case of them being applied to circuits that are redundant in nature. Consider that redundant circuits run through the wire lifting procedure would be subject to one of the following two outcomes, assuming the same number of wires are lifted from each circuit:

1. Each redundant portion of the circuit will be identical to one another after the wire lifting procedure.
2. The *k-security* of the circuit will be greatly reduced.

For example, Figure 2 shows the graph representation of a redundant circuit. This is Graph 1 taken from Figure 1 duplicated through a pipeline stage. It has two redundant portions represented by nodes `1-6` and nodes `7-12`. If we remove the inputs (`A`, `B` and `C`) and the outputs (`D`, `E` and `F`) as well as the pipeline stage and allow the circuit to undergo the wire lifting procedure Figure 3 and Figure 4 are two possible results. Figure 3 aligns with point 1 from above. The two redundant portions of the design have had identical wires lifted. If a single redundant portion was examined by itself, it would look identical to Graph 2 taken from Figure 1 and would have a *k-security* of 2. However, the

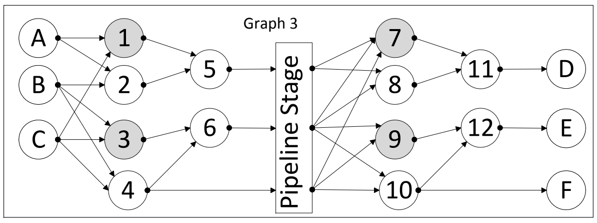

Fig. 2 An example of a redundant circuit with two redundant portions and a pipeline stage separating the two.

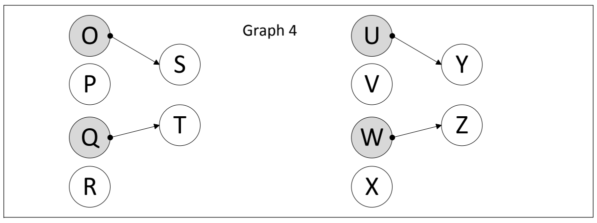

Fig. 3 A redundant circuit which has undergone the wire lifting procedure and each redundant portion is identical to each other. The *k-security* of the circuit is 4.

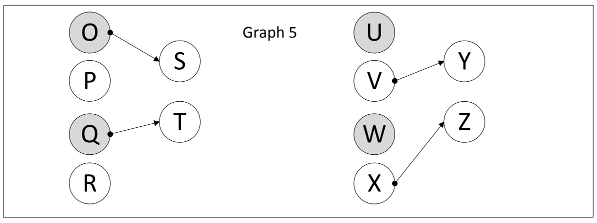

Fig. 4 A redundant circuit which has undergone the wire lifting procedure and the two redundant portions are not identical to each other. The *k-security* of this circuit is 2.

circuit as a whole has a *k-security* of 4, i.e., each subgraph is indistinguishable from three other subgraphs.

Figure 4 aligns with point 2 from above. In this case, after the wire lifting procedure the two redundant portions of the circuit are not identical to one another but have the same number of wires lifted as Figure 3. Each redundant portion of the circuit, when examined independently, has a *k-security* of two and the circuit as a whole also has a *k-security* of 2. The only way to increase the *k-security* of Figure 4 to four would be to remove the remaining wires. Minimizing the number of wires lifted is important because as the number of lifted wires increases, so increases the power consumption, delay, and area of a circuit [11].

The above illustrates the security concern that redundant circuits introduce: if an attacker wished to modify the circuit represented by Graph 3 in Figure 2 so that the output D was always held to `0`, they would need to attack node 11. Similarly, the DES circuit example in [11] suggested that a successful Trojan would need 5× the number of gates as the *k-security* of the node to be attacked. In fact, this could be done with exactly the number of gates equal to the *k-security* of the node to be attacked. In Figure 3 an `AND` gate with one input held to `0` once triggered, would be attached to nodes `S`, `T`, `Y`, and `Z`. This would guarantee that output `D` was held to zero, and would require exactly four `AND` gates.

4 Weaknesses of Split Manufacturing Obfuscation on Redundant Circuitry

The following section describes the threat model, the specific DES circuit that was attacked, and the attacks which demonstrate that the wire lifting procedure set forth in [11] does not provide the the level of security that it claims because of the redundant nature of the DES circuit.

4.1 Threat Model

We use the same threat model as the authors of [11]. We assume that the Trojan is inserted during the fabrication phase. In addition, the attacker has full knowledge of the original circuit. Also, a trusted party has performed the wire lifting procedure, as described earlier, and manufactured the trusted tier. The attacker does not have a knowledge of the results of the wire lifting procedure. Hence, if the attacker intends to change the behavior of the circuit, they can only do so with a one in k chance of success, where k is the *k-security* of the circuit. This is because the attacker will not be able to differentiate any gate between itself and $k-1$ other gates.

We assume a pipelined DES circuit on an application-specific integrated circuit (ASIC), as explained in Section 4.2. The attacker is able to supply the device with plaintext challenges of their choosing and observe the cipher text outputs. The attacker is also able to trigger the Trojan, thus allowing them to learn the secret key.

4.2 The DES Circuit

We begin by discussing the DES circuit used in the original work. The circuit from [11] is described as having approximately 35,000 logic gates, which, based upon an examination of publicly available DES cores, matches well with a pipelined implementation of DES, and so we assumed a pipelined architecture.

DES has 16 rounds of logic that are identical to one another [24], hence it is a redundant circuit. DES was used in [11] as a demonstration of the difficulty an attacker would have implanting a successful hardware

Trojan into a design that had undergone the wire lifting procedure. Per [11] a *k-security* of 16 is achieved by simply removing the interconnects between the rounds; i.e., each round is indistinguishable from any other round in the circuit. Also, as stated in [11], after the wire lifting procedure the final circuit is 64-secure with the bit under attack, the LSB of the fourteenth round, being in fact 256 secure.

As this is a fully pipelined design, a cipher text appears at the output on each clock cycle, after the initial delay of filling the pipeline. Each of the 16 rounds is assumed to be operating on a different encryption on the same clock cycle, so there are 16 different plaintexts being encrypted on each clock cycle, provided the pipeline remains full.

4.3 Attack Background

In order to introduce the attacks we propose, and to explain the only attacks that Imeson et al. claimed were possible, let us return to Figure 2, an example circuit with two redundant portions, and Figure 3, which depicts a circuit that has undergone the wire lifting procedure. This example does not show a DES circuit but we use it for the purposes of illustration. After the wire lifting procedure has taken place, nodes `S`, `T`, `Y`, and `Z` are indistinguishable from node `11`. If an attacker was interested in causing a fault on node `11`, from Figure 2, Imeson et al. claim that there are two ways to do this after the wire lifting procedure. Either an attacker must choose one of the indistinguishable nodes to attack, and accept a low chance of success (in this case they have a 1/4 chance of success), or attack each of the nodes, one at a time. The problem with attacking each of the indistinguishable nodes one at a time is that it creates a very large Trojan because of the large multiplexer that would be required to select the gate under attack.

For this example we would propose that all four of the indistinguishable nodes be attacked at the same time. Because the redundant portions are separated by a pipeline stage, the faults induced on the nodes in the first stage have no effect on the outcome of the injected fault on second stage. We will show that this style of attack would have a higher probability of success than the small Trojan of [11] and be much smaller than their large Trojan.

Let us extend this example to DES, which was used as an example of a large circuit for which it is claimed the wire lifting procedure would prevent a Trojan from being successfully implanted. A known weakness of DES to fault injection, which was discussed in Section 3.2, was identified as a potential attack vector for a Trojan. Specifically, a Trojan needs to target the LSB of the fourteenth round: either the attacker could choose one of the 256 indistinguishable possibilities and have a 1/256 chance of success, or attack each of the options one at a time in a multiplexed attack which would yield a larger Trojan. However, we suggest that an attacker might attack each of the 256 bits all at once, holding them at zero simultaneously for a single clock cycle. This new attack would have two effects.

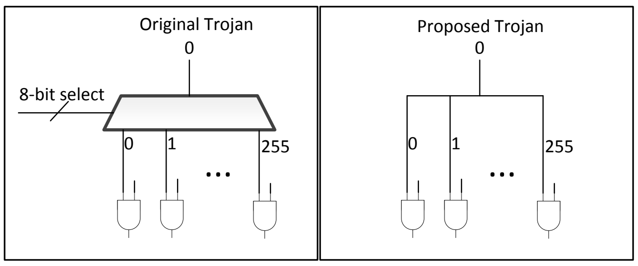

Fig. 5 The figure on the left shows the original Trojan which requires 1280 gates. The figure on the right shows a smarter Trojan which requires 256 gates, numbered 0 to 255.

The first effect is that it would hold the LSB of round fourteen to zero for a particular plaintext. It would also hold fifteen other bits to zero for the fourteenth round as the LSB of the output is 16-secure within the round. The other effect is that each of the other rounds will have 16 bits that are also held to zero. However, for a pipelined design the bits held to zero in the other rounds will have no effect on the cipher text output in question. The other plaintexts with induced errors are be discarded.

This new design for the Trojan is compared to the original in Figure 5. The removal of the multiplexer decreases the size of the Trojan dramatically, from 1280 gates to 256.

4.4 Attack Implementation

We discuss two attacks against DES. A pipelined implementation of DES was obtained from opencores.org [26], in order to be unbiased, and used for these attacks. The circuit was unmodified with the exception of inserting the hardware Trojan into the design. As the wire lifting procedure set forth in [11] did not converge, we used several pessimistic assumptions to hinder the attacker in an effort to be fair and unbiased.

In the first attack we induced a fault at the LSB output of the fourteenth round along with fifteen other random locations in the round. This is pessimistic because any bit in the round is a candidate for the fifteen fault locations chosen at random. In reality, only a similar subcircuit should be eligible as it would need to be logically identical to the LSB output of the fourteenth round for the wire lifting procedure to leave them indistinguishable from each other. In the second attack the circuit is examined as a whole and so the LSB output of the fourteenth round has a fault induced on it, as well as 255 other, randomly selected locations throughout the circuit. These self-imposed pessimistic assumptions make it very difficult for the attacker to succeed. Both of these attacks were carried out using Verilog simulations.

The simulations were run as follows. We randomly selected the bits to be held to zero along with the LSB of the fourteenth round. These bits were not limited to portions of the circuit that may be indistinguishable from the LSB of the fourteenth round. This is a very pessimistic approach for the attacker because it decreases the chance of being able to recover the key given that more bits are corrupted. Each attack was simulated 10,000 times with each simulation, again, selecting new random locations in the circuit for the Trojan to hold to 0, along with the LSB of the output of the fourteenth round. This was done in lieu of the wire lifting procedure.

4.4.1 Attack One: Limit Scope to the Round Fourteen

The first attack entails attacking a single round in the unbiased DES circuit. If we assume that the entire circuit is 256-secure and there are 16 identical rounds, then if the LSB has 255 other gates that look exactly like it after the wire lifting procedure then the fourteenth round would have fifteen other gates that are indistinguishable from it. This amounts to 16 bits per round. (This assumption is based on the statement from page twelve of [11] which states, "We note that a security level of 16 is obtained in the first few rounds of partitioning by removing 13% of the wires, i.e., all wires that lie between successive DES rounds." [11]) Therefore, in addition to the `AND` gate placed on the LSB of the fourteenth round to hold it to zero we also randomly selected fifteen other locations where a value would be held to zero by an `AND` gate triggered at the same instant as the gate tying the LSB to zero.

We selected fifteen random locations to provide an unbiased simulation of what gates an attacker might be faced with in a 16-secure round. Only fifteen random locations were needed in the fourteenth round because the other 240 $(255-15)$ other locations that would be indistinguishable from the LSB of the fourteenth round are corrupted as well, however, they are in different rounds which cause corruption to different plaintexts being encrypted. Sixteen total corrupted plaintexts would result but we are only interested in the corrupted plaintext resulting from the corruption in the fourteenth round. The additional cipher texts are discarded.

4.4.2 Attack Two: Allow any bit to be corrupted

In an effort to expand our findings, we designed a second attack. Although we felt that our assumptions used in the first attack were sound, in an attempt to be unbiased, we created a second, more restrictive attack to confirm the weakness of the wire lifting procedure. If we had been able to successfully complete the wire lifting procedure this would not have been necessary, as we could have proven our previous assumptions.

This second attack extended the first attack to consider the circuit as a whole. As the LSB of the fourteenth round is said to be 256-secure, we randomly selected 255 bits from anywhere in the circuit and held those bits to zero in the same clock cycle that we held the LSB of the fourteenth round to zero. Note, there was other logic in the pipelined DES circuit that was outside of the round logic, such as the key scheduler. The setup for this attack included all logic in the design, not just the logic found within the 16 rounds.

Each simulation represents a single implementation of possible Trojan logic. The actual Trojan circuit would contain 256 `AND` gates to inject a fault onto each gate indistinguishable from the LSB of the fourteenth round. These simulations helped us determine the likelihood of success of an attack on a DES circuit that has undergone the wire lifting procedure.

4.5 Attack Results

For each of the two attacks the plaintext input remained constant to reduce the number of variables. Both the corrupted and uncorrupted ciphertext outputs were collected from Verilog simulations. These outputs were analyzed using the fault injection analysis methods described in [3] and expanded upon in [24]. A C program was written to automatically implement these methods to determine whether or not the secret key could be identified.

These methods created several guesses for the round key of the fifteenth round. Each guess contained 48 of the needed 56 bits of the secret key. Then the remaining 256 possibilities for the remaining eight bits were exhaustively searched through a software DES implementation. If the cipher text resulting from the key guess matched the uncorrupted cipher text output then the key that was used to create it was indeed the original key.

For the first attack Figure 6 displays the results. Out of the 10,000 simulations run, there were 7,554 that

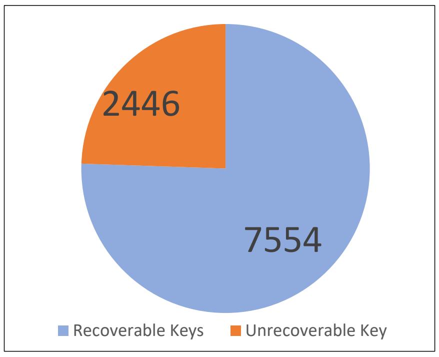

Fig. 6 Attack 1 results showing that on a pipelined DES circuit 7,554 out of 10,000 keys were recoverable assuming that the faults in all other rounds can be disregarded. This proposed Trojan is 5x smaller than the large Trojan proposed by [11], and has a success rate of 191x their small Trojan.

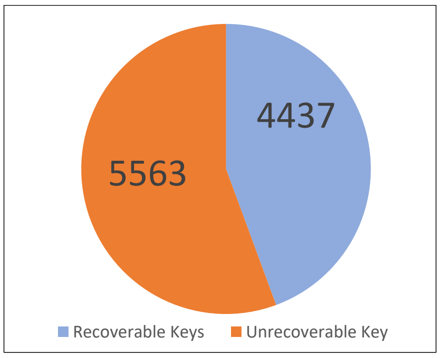

Fig. 7 Attack 2 results showing that on a pipelined DES circuit 4,437 out of 10,000 keys were recoverable. This attack forced a very pessimistic view for the attacker, which would not be reasonable. It allowed any bit in the circuit to be corrupted by the Trojan. This pessimistic view was given to show that even given the most possible pessimistic assumptions the proposed Trojan was still 5x smaller than the large Trojan proposed by [11], and has a success rate of 112x their small Trojan. If the wire lifting procedure had converged, this pessimistic view would not have been necessary.

resulted in a key that was easily recoverable. The results of the second attack can be seen in Figure 7. Out of the 10,000 simulations run there were 4,437 recoverable keys. It should be noted that the completely random nature by which we selected the 255 points in the second attack actually biased key recovery towards failure. In an actual obfuscated circuit the LSB of the fourteenth round could not have been confused with every other bit in the entire design. Still, our observed success rate of approximately 75% (191/256) and 44% (112/256) is far better than the 1/256, or 0.4% success rate that was discussed in [11].

The other alternative [11] offered has a guaranteed success rate but requires a Trojan that has 1280 gates (excluding trigger logic). While the an acceptable Trojan size was not discussed in the original work, the Trojan proposed here would be only 256 gates excluding trigger logic. This Trojan is 5x smaller in size, which is again a substantial improvement.

4.6 Discussion

The results above show a significant improvement using the metrics that the authors of [11] used. However, these results would have been even better given realistic assumptions not utilized because of our inability to complete the wire lifting procedure. For example, when the actual architecture of the circuit is considered, the bits that could be indistinguishable from the LSB of the fourteenth round are output bits 1–15 of the same round. In this case instead of only inducing a known fault on a single bit, we would be able to induce it in 16 known bits in the fourteenth round. The 240 bits in the other rounds that would be indistinguishable from the LSB of the fourteenth round are output bits 0-15 of each of the other fifteen rounds, and these would have no effect on the outcome of the attack. If this assumption was allowed to be made, the success rate of the Trojan would be 100%. Fault injection analysis teaches that the greater the number of known bits that have a fault induced on them, the easier it is to recover the key [24].

5 Weaknesses Extended to AES

Hardware Trojans can also be created to make AES circuits vulnerable, even after the wire lifting procedure has been performed. The example of AES being vulnerable is important because it extends the previous attack to another class of cryptographic algorithms (those based on SPN). For our purposes, 128-bit AES was considered but the attack can be extended to 192 and 256-bit AES, as well. It must be reiterated that because the publicly available wire lifting procedure failed to successfully terminate, several pessimistic assumptions were made.

5.1 AES Attack Background

The original proposal for AES was submitted to the National Institute of Standards and Technology in 1999 describes the Rijndael algorithm, a symmetric block cipher [5]. It was adopted as a standard in 2001 [6]. The standard allows for the processing of 128 bit data blocks with an option of key sizes of 128, 192 or 256 bits. The 128-bit data is organized into a 2D array of 4x4 bytes called the State. For a 128-bit key operating on 128-bit

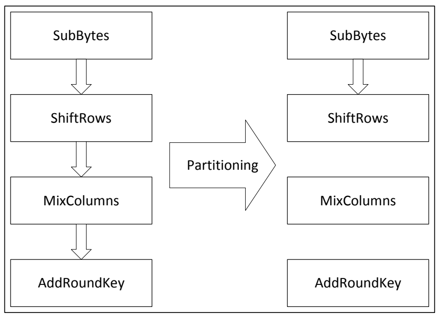

Fig. 8 After partitioning the rounds to remove wires between their components, there would be nine identical MixColumns circuits, ten identical circuits of SubBytes followed by ShiftRows, and ten AddRoundKey circuits.

data, AES has ten rounds. Rounds 1-9 are all identical with their State manipulation steps as follows [6]:

1. Perform byte substitutions.
2. Shift the rows of the matrix.
3. Perform the mix column transformation.
4. Add the round key to the result.

The final round, round ten performs steps 1, 2 and 4, but leaves out step 3, the mix column transformation. This is an important distinction from DES which has 16 identical rounds [24]. This difference is important because in [11] the idea of manually partitioning the circuit and removing the wires between the rounds is the first step of the procedure, which creates a *k-security* of 16. With AES if the connections between the rounds are broken, no such *k-security* is achieved because the final round is not identical to the others. With this in mind, we would suggest partitioning the AES circuit in a way that would allow for the greatest *k-security* as was done in the example of DES. We submit that the best case scenario is for removing the wires between all rounds, as well as between steps 2 and 3 and between steps 3 and 4 before performing the wire lifting procedure, as seen in Figure 8. This is important to do as otherwise the final round could be easily attacked, given that it would have a recognizably distinct footprint from the other rounds.

After this partitioning procedure is completed the final round circuit of SubBytes to ShiftRows, as well as the AddRoundKey circuit, is ten-secure, as there would be nine other rounds in which those identical circuits existed. We would now pass the partitions through the wire lifting procedure with a goal of having each circuit be x-secure. Note that in order to add anonymity to the final round, we have isolated the AddRoundKey circuit which is by default 128-secure in each round (this is because that circuit is made up entirely of 128 `XOR` gates). The whole circuit, then, is in fact 1280-secure. There are 1279 other gates in the circuit which cannot be distinguished from a particular `XOR` gate in the final AddRoundKey circuit. 127 of these 1279 gates also exist in the final AddRoundKey circuit. We make these assumptions about the final *k-security* of the circuit as if the wire lifting procedure had run efficiently.

5.2 AES Attack Outline

In order to recover the 128-bit key of AES, we propose to attack the circuit in the final round during the AddRoundKey step (when the round key is added in). That is, if we can implant a hardware Trojan that can cause a fault that holds a bit to `0` in at least 64 of the 128 bits that are `XOR'd` with the round key, those bits of the round key can be revealed. The final bits of the key (up to 64 bits) can be brute forced by an exhaustive search until the full key is recovered. After the attack the remaining bits of the round key would be guessed and the round key would be propagated through the AES key schedule in reverse to reveal the key. That key would be used to encrypt a known plaintext-ciphertext pair. If the ciphertext encrypted under the guessed key matched the original ciphertext then the guessed key is correct. If not, then the process starts over by guessing the remaining bits in the final round key, again.

This attack uses the properties of the `XOR` operation: `A XOR B = B` where `A = 0`. Any bit of the final round key will be revealed if it is `XOR'd` with `0`. If our hardware Trojan affects random bits that are `XOR'd` with the final round key, we do not need to attack specific bits. In fact, we do not even need to know how many bits were attacked, we only need enough plaintexts to recover all the bits. The following is an illustrative example.

Let us consider the least significant byte of the final step, adding the round key, of the final round of an AES encryption. Assume that two bits of this byte are held low when a hardware Trojan is active but it is not known which bits are affected by the Trojan. Given that enough ciphertext pairs `(C,C')`, where `C` is a correctly encrypted ciphertext and `C'` is a ciphertext encrypted while the hardware Trojan was active, and they are both the encryption of the same plaintext, we can determine which bits are associated with the Trojan and the value of those bits of the round key. For example, assume that the least significant byte of the round key (from here on referred to as "the key") is `10101010` and the least significant byte of the State being `XOR'd` with the key is `11110000`. In this case, `C = 01011010` and `C' = 01001010`. The errored byte E is calculated by `C XOR C'`, which in this case yields `00010000`, where a `1` indicates a bit that was affected by the Trojan.

Now, using E as a mask over `C'` to find the known values of the key, which are indicated in bold, `010`**`0`**`101`.

We see that bit four (starting with zero on the right) of the round key is `0` and we have recovered one of the two bits. Continuing this example, after several more ciphertext pairs, in which E = `00000000`, we find the `(C,C')` pair `(00101101, 00101111)` and E = `00000010`. Again using E as a mask over `C'` to select the revealed portion of the key, this indicates that bit one of the key is `1`. At this point we have recovered both bits of the round key that this Trojan allows; i.e., we discovered that the round key looks like `---0--1-`. More bits affected by the Trojan would have revealed more key bits.

The attack above is similar to the attack on the DES circuit in that the Trojan consists of `AND` gates designed to hold the bits they affect to `0` when activated. Like the attack on the DES circuit, a pipelined AES implementation is used. The entire Trojan is activated at the same time, for a single clock cycle, so if gates of the Trojan affect bits in rounds besides the final round, they will not change the ciphertext output that we are concerned with. Only the gates of the Trojan that lie within the final round will cause any change to that ciphertext.

5.3 AES Attack Implementation

A pipelined implementation of AES was obtained from opencores.org [9], in order to start with an unbiased implementation. This design also came with a test bench that was utilized for simulation. The circuit and test bench were modified only to add the hardware Trojan—in no other way was the circuit tampered with.

The *k-security* of any of the `XOR` gates in the final round during the AddRoundKey step is 128 with respect to that round and 1280 with respect to the entire circuit. If our intention was to attack a specific `XOR` gate we would need a hardware Trojan to contain 1280 `AND` gates to be sure the crucial gate was attacked. Instead, we only want enough gates to be attacked so that we can recover the key. We claim that at least 64 of these `XOR` gates in the final round must be attacked, leaving up to 64 bits of the key to be exhaustively searched. In order to decide how large to make the Trojan, we simulated different Trojan sizes 10,000 times. For each simulation we kept the number of Trojan bits constant but randomly selected their locations. This is akin to seeing the netlist that an attacker has access to but not knowing which of the 1280 `XOR` gates fall within the final round. Instead of attacking each gate the attacker would select a number of them at random to attack. These simulations are used to calculate the probability of success with varying Trojan sizes. The sizes of Trojans simulated were 640 bits, 700 bits and 800 bits.

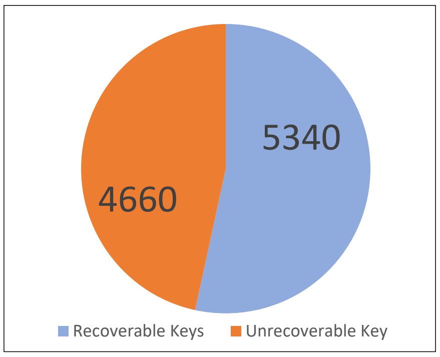

Fig. 9 Results for the 640 gate Trojan attacking AES are given. Out of 10,000 simulations, 5,340 resulted in recoverable keys. This success rate can be improved by increasing the size of the Trojan.

A Python program was written to generate the 10,000 locations of each bit of the Trojan for each of the three Trojan sizes. The Verilog code was modified to incorporate these locations and the resulting circuits were simulated with the `(C,C')` pairs being written to a file. Those `(C,C')` pairs were then analyzed by another Python program to determine the number of bits of the round key that were discovered as well as how many pairs were needed to find those bits.

5.3.1 640-bit Trojan Results

The 640-bit hardware Trojan was selected as a starting point as it was half as large as the 1280 which would affect all gates. The results show that in 5,340 of the 10,000 simulations at least 64-bits of the key were recovered. The maximum number of round key bits recovered was 85 and the minimum number recovered was 43. The probability of success for this Trojan size was 53.4%. All bits of the round key were recovered using at most six `(C,C')` pairs. These results can be seen in Figure 9.

5.3.2 700-bit Trojan Results

A small increase in Trojan size revealed an increased probability of success: 8,938 simulations resulted in 64 bits or more of the round key being revealed. This is an 89.38% chance of success with this Trojan size. The maximum number of round key bits recovered was 92 and the minimum was 52. These bits were again found using at most six `(C,C')` pairs. These results can be seen in Figure 10.

5.3.3 800-bit Trojan Results

This final simulation revealed that at least 64 bits of the round key were recovered in 9,992 of the simulations which is a 99.92% chance of successfully recov-

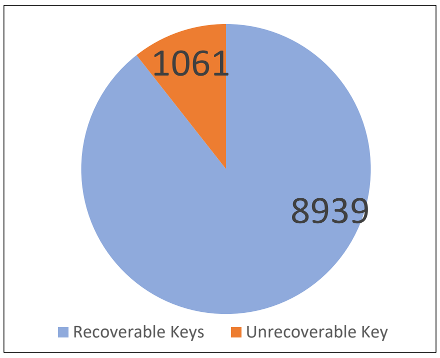

Fig. 10 Results for the 700 gate Trojan attacking AES are given. By increasing the Trojan size by 60 gates, or just over 10% we were able to increase the recoverable keys by 67% over the 640 gate Trojan. Out of 10,000 simulations, 8,938 resulted in recoverable keys.

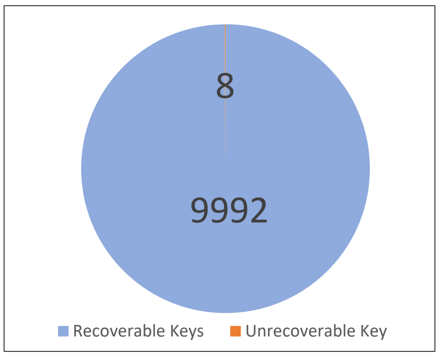

Fig. 11 Results for the 800 gate Trojan attacking AES are given. Out of the 10,000 simulations run, 9,992 had recoverable keys. This is nearly a 100% success rate. This would be the maximum Trojan size needed for a highly successful Trojan.

ering the key. The minimum number of bits recovered in this simulation was 61 and the maximum number of bits was 101. The maximum number of bits recovered would only leave 27 bits of search space or 134,217,728 combinations, required to discover the full key. This is a problem a typical modern-day personal computer could easily solve. The maximum number of `(C,C')` pairs required to recover each bit was again six. These results can be seen in Figure 11.

5.4 Method applied to the DES circuit

A similar Trojan that was used against the AES circuit could be modified to attack the DES circuit previously described. In this scenario the hardware Trojan would attack the entire left hand output of the fourteenth round, holding it to zero. This would require 32 `AND` gates per round for a total of 512 `AND` gates. This would allow us to discover the round key for the 16th round with 100% certainty using only a single plaintext.

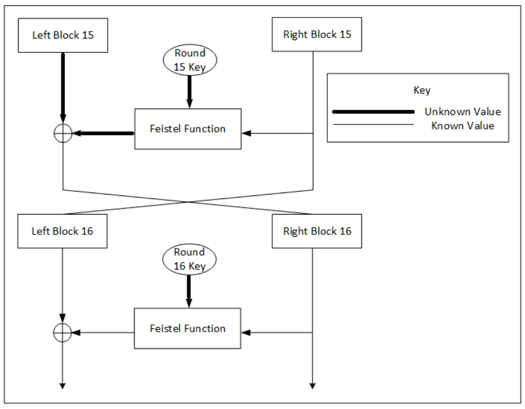

Fig. 12 Rounds fifteen and 16 of DES are shown. It is illustrated that if a hardware Trojan caused the left block output of round fourteen to be zeros then the only unknown for round 16 is the round key.

This works because if the left block output of the fourteenth round is held to zeros, then the right block input to the fifteenth round would also be zeros and the left block input to the 16th round would be zeros, as well. Both the right and left block outputs of the 16th round are known because they can be reversed through the final permutation to reveal them. The right block input to the 16th round would also be known because it is equal to the left output. This is illustrated in Figure 12. Knowing all inputs and outputs to the round, the only unknown would be the round key which could easily be determined by going through the algorithm in reverse. Upon discovering the round key, the remaining eight bits of the secret key could be discovered by an exhaustive search, as described earlier.

This Trojan to attack DES would be 512 gates large, which is half the size of the original Trojan suggested by Imeson et al., and it would have a 100% success rate.

6 Conclusion and Future Work

We have demonstrated that the obfuscation methods set forth in [11] do not provide the claimed security for Feistel or SPN structured circuits. This is because these circuits contain highly redundant logic, which makes them vulnerable to hardware Trojan attacks even after the wire lifting procedure has been carried out, due to the fact that even if a desired gate cannot be attacked directly, there is enough redundancy in these types of circuits to allow for an attack to be successful. In these cases, *k-security* gives a false sense of how secure the circuit is because multiple gates in a redundant, pipelined circuit can be attacked simultaneously without the extraneous gates affecting the outcome of the attack, as they are attacked during a different pipeline stage.

Our work shows that it is difficult to prove that any method of split circuit manufacturing has the same benefits for arbitrary classes of circuits. It is therefore beneficial when a class of circuits can be identified as being an exception to the rule. Future split manufacturing approaches must consider redundant circuits, specifically. That is, once a new method is identified it must be tested against redundant circuits to see if the claims of security still hold for those types of circuits.

Future work would include investigating other types of redundant circuits in order to determine whether cryptographic circuits are the only types of highly redundant circuits for which the wire lifting procedure does not provide the asserted amount of security. Other classes of circuits may also be identified as not being subject to the claims of increased security provided by split manufacturing.

Future work should also include investigating advances made to the original paper, by other authors to ensure the techniques described here continue to allow Trojan implantation. There is continuing research on improving split manufacturing to further obfuscate the original design by inserting dummy wires and cells [14]. This would make the methods presented here more challenging, but still achievable, with perhaps slightly larger Trojans.

References

1. Baigneres, T., Vaudenay, S.: Proving the security of aes substitution-permutation network. In: International Workshop on Selected Areas in Cryptography, pp. 65–81. Springer (2005)
2. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE **100**(11), 3056–3076 (2012)
3. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Annual International Cryptology Conference, pp. 513–525. Springer (1997)
4. Committee on Armed Services, United States Senate: Inquiry into counterfeit electronic parts in the department of defense supply chain (1999). URL http://www.armed-services.senate.gov/Publications/Counterfeit%20Electronic%20Parts.pdf
5. Daemen, J., Rijmen, V.: Aes proposal: Rijndael (1999)
6. Daemen, J., Rijmen, V.: Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication **197** (2001)
7. Girija, R., Singh, H.: A new substitution-permutation network cipher using walsh hadamard transform. In: Computing and Communication Technologies for Smart Nation (IC3TSN), 2017 International Conference on, pp. 168–172. IEEE (2017)
8. Hasegawa, K., Oya, M., Yanagisawa, M., Togawa, N.: Hardware trojans classification for gate-level netlists based on machine learning. In: On-Line Testing and Robust System Design (IOLTS), 2016 IEEE 22nd International Symposium on, pp. 203–206. IEEE (2016)
9. Hsing, H.: tiny_aes (2015). URL https://opencores.org/project/tiny_aes
10. Imeson, F.: circuit_security (2017). URL https://github.com/fcimeson/circuit_security
11. Imeson, F., Emtenan, A., Garg, S., Tripunitara, M.: Securing computer hardware using 3d integrated circuit ({IC}) technology and split manufacturing for obfuscation. In: Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13), pp. 495–510 (2013)
12. Johnson, A.P., Patranabis, S., Chakraborty, R.S., Mukhopadhyay, D.: Remote dynamic clock reconfiguration based attacks on internet of things applications. In: Digital System Design (DSD), 2016 Euromicro Conference on, pp. 431–438. IEEE (2016)
13. Kim, C.H., Quisquater, J.J.: Faults, injection methods, and fault attacks. IEEE Design & Test of Computers **24**(6), 544–545 (2007)
14. Li, M., Yu, B., Lin, Y., Xu, X., Li, W., Pan, D.Z.: A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **38**(9), 1585–1598 (2018)
15. Maity, G., Bhaumik, J., Kundu, A.: A new spn type architecture to strengthen block cipher against fault attack. IJ Network Security **20**(3), 455–462 (2018)
16. Malekpour, A., Ragel, R., Ignjatovic, A., Parameswaran, S.: Trojanguard: Simple and effective hardware trojan mitigation techniques for pipelined mpsocs. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 19. ACM (2017)
17. Peng, J., Tan, C.H., Wang, Q., Gao, J., Kan, H.: More new classes of differentially 4-uniform permutations with good cryptographic properties. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(6), 945–952 (2018)
18. Rajendran, J.J., Sinanoglu, O., Karri, R.: Is split manufacturing secure? In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1259–1264. EDA Consortium (2013)
19. Salmani, H.: Hardware trojan attacks and countermeasures. In: Fundamentals of IP and SoC Security, pp. 247–276. Springer (2017)
20. Salmani, H., Tehranipoor, M., Plusquellic, J.: New design strategy for improving hardware trojan detection and reducing trojan activation time. In: Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on, pp. 66–73. IEEE (2009)
21. Samimi, M.S., Aerabi, E., Kazemi, Z., Fazeli, M., Patooghy, A.: Hardware enlightening: No where to hide your hardware trojans! In: On-Line Testing and Robust System Design (IOLTS), 2016 IEEE 22nd International Symposium on, pp. 251–256. IEEE (2016)
22. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher clefia. In: International Workshop on Fast Software Encryption, pp. 181–195. Springer (2007)
23. Tehranipoor, M., Koushanfar, F.: A survey of hardware trojan taxonomy and detection. IEEE design & test of computers **27**(1) (2010)
24. Tunstall, M.J.M.: Fault Analysis in Cryptography. Springer (2012)
25. U.S. department of commerce bureau of industry and security office of technology evaluation: Defense industrial base assessment: Counterfeit electronics (1999). URL http://www.bis.doc.gov/defenseindustrialbaseprograms/

osies/defmarketresearchrpts/final_counterfeit_electronics_report.pdf
26. Usselmann, R.: DES/Triple DES IP Cores (2009). URL http://opencores.org/project/des
27. Vaidyanathan, K., Das, B.P., Pileggi, L.: Detecting reliability attacks during split fabrication using test-only beol stack. In: Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE, pp. 1–6. IEEE (2014)
28. Vaidyanathan, K., Das, B.P., Sumbul, E., Liu, R., Pileggi, L.: Building trusted ics using split fabrication. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST), pp. 1–6. IEEE (2014)
29. Vaidyanathan, K., Liu, R., Sumbul, E., Zhu, Q., Franchetti, F., Pileggi, L.: Efficient and secure intellectual property (ip) design with split fabrication. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST)n, pp. 13–18. IEEE (2014)
30. Wang, M.T.C.: Introduction to Hardware Security and Trust. Springer, 233 Spring Street, New York, NY 10013 (2012)
31. Wang, Y., Chen, P., Hu, J., Li, G., Rajendran, J.: The cat and mouse in split manufacturing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **26**(5), 805–817 (2018)
32. Xie, Y., Bao, C., Srivastava, A.: Security-aware design flow for 2.5 d ic technology. In: Proceedings of the 5th International Workshop on Trustworthy Embedded Devices, pp. 31–38. ACM (2015)
33. Yoshimura, M., Bouyashiki, T., Hosokawa, T.: A hardware trojan circuit detection method using activation sequence generations. In: Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on, pp. 221–222. IEEE (2017)