Red Team vs. Blue Team Hardware Trojan Analysis*

Detection of a Hardware Trojan on an Actual ASIC

Michael Muehlberghuber
ETH Zurich
Gloriastrasse 35
8092 Zurich, Switzerland
mbgh@iis.ee.ethz.ch

Frank K. Gürkaynak
ETH Zurich
Gloriastrasse 35
8092 Zurich, Switzerland
kgf@ee.ethz.ch

Thomas Korak
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria
thomas.korak@iaik.tugraz.at

Philipp Dunst
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria
p.dunst@student.tugraz.at

Michael Hutter
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria
michael.hutter@iaik.tugraz.at

ABSTRACT

We infiltrate the ASIC development chain by inserting a small denial-of-service (DoS) hardware Trojan at the fabrication design phase into an existing VLSI circuit, thereby simulating an adversary at a semiconductor foundry. Both the genuine and the altered ASICs have been fabricated using a 180 nm CMOS process. The Trojan circuit adds an overhead of only 0.5 % to the original design. In order to detect the hardware Trojan, we perform side-channel analyses and apply IC-fingerprinting techniques using templates, principal component analysis (PCA), and support vector machines (SVMs). As a result, we were able to successfully identify and classify all infected ASICs from non-infected ones. To the best of our knowledge, this is the first hardware Trojan manufactured as an ASIC and has successfully been analyzed using side channels.

Categories and Subject Descriptors

B.8.1 [**Hardware Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; C.4 [**Performance of Systems**]: Reliability, availability, and serviceability

General Terms

Security

HASP '13, June 23-24 2013, Tel-Aviv, Israel

*This work has been supported in part by the Swiss Commission for Technology and Innovation (CTI) under project number 13044.1 PFES-ES. This work has been also supported in part by the European Commission through the ICT program under contract ICTSEC- 2009-5-258754 (Tamper Resistant Sensor Node - TAMPRES).

Keywords

Hardware Trojan, ASIC fabrication, side-channel analysis, PCA, SVM

1. INTRODUCTION

Modern integrated circuit (IC) fabrication relies on an ever increasing set of active participants to manage both costs and complexity. More often than not, large parts of the design are outsourced to specialist teams all around the world, pre-designed intellectual property (IP) blocks from different vendors are added to the design, complex electronic design automation software is used in the design flow for synthesis and analysis purposes, and finally in many cases a specialized semiconductor foundry is used for the actual manufacturing of the ICs.

The complex development chain used for the application-specific integrated circuit (ASIC) provides a skilled adversary various possibilities for intrusion and therefore, an adversary could manage to insert a (relatively small) circuit that can remain undetected during the design process. Such a hardware Trojan [4] would then be manufactured together with the actual circuit, and could be used for different malicious purposes such as denial-of-service attacks and/or leaking sensitive information.

The complexity of ASICs continues to increase exponentially as dictated by the Moore's Law. As a consequence, the opportunities for an attacker to insert a hardware Trojan continue to grow at the same time. This is due to the following factors:

1. The proportion of a hardware Trojan to the actual circuit constantly decreases. Detecting an unwanted circuit which is smaller than one millionth of the actual circuit becomes more and more difficult.

2. The cost of developing ASICs is only justified for systems where the absolute performance (speed, area, security) are of paramount importance or for high-volume manufacturing where millions of ASICs are manufactured. Both cases are attractive targets for adversaries, as ASICs often constitute either the most important part of a system, or are widely deployed and thereby compromise a large number of systems.

DOI: http://doi.acm.org/10.1145/2487726.2487727.

3. As the expected harm caused by a compromised system is very high, adversaries can afford to invest significant sums into designing and inserting hardware Trojans.

At the same time, ASIC designers face significant challenges to verify the correct functionality of their own (correct) circuits properly. Even the most modern verification flows do not stand much chance to detect hardware Trojans inserted by determined adversaries. That is why, hardware Trojan detection (e.g., see [8, 9, 13] for further information) has attracted significant interest in recent years. Given sufficient time and resources, all hardware Trojans could be detected by careful visual inspection. However, this takes a long time, and especially for modern manufacturing processes is usually invasive and/or expensive. Therefore, there is a pressing need to develop and refine cost-efficient, hardware Trojan detection systems.

In this work, we present the results of a study where a small hardware Trojan (less than 200 gate equivalents (GE)) was designed and inserted in an actual ASIC of medium complexity (around 40 kGE) by the *red team* at layout-mask level. The *red team* had little knowledge of the consequences on the target design called Chameleon prior to the insertion, and had to finish the work within two working days. Both the original ASIC Chameleon and the compromised ASIC, called Chipit, were manufactured on the same run using the UMC 180 nm CMOS technology. A mixed group of manufactured ASICs were then made available to the *blue team*, which was able to correctly identify all compromised chips containing the hardware Trojan using methods based on measuring power related side channels.

Our Contributions

In this paper, we contribute in several ways. We can list the main points as follows:

1. We inserted a hardware Trojan into a real ASIC design and successfully performed side-channel analysis to (automatically) detect them. To the best of our knowledge, this is the first time a hardware Trojan has been inserted in a real ASIC and has reliably been detected.

2. In contrast to related work, we followed a very realistic scenario where we inserted a hardware Trojan at the mask-layout level. This is in fact the last stage before production in a fab.

3. The inserted Trojan has a very small footprint, i.e., less than 0.5 % of the total chip area. We therefore answer the open question if such small Trojans can be reliably detected using side-channel information.

4. The inserted Trojan is *always active* and therefore has no dormant state. This makes it very difficult to detect as there are no transient activation signals that can be identified by side-channel analysis.

5. We performed our evaluations on eight different ASICs: three chips contain a hardware Trojan, and five chips do not. We further followed a real-world attack scenario where one research group (red team) designed and inserted the Trojan and another (independent) research group performed the analyses and Trojan detection without any knowledge of the design and classification.

6. As a Trojan detection method, we apply several side-channel analyses techniques proposed in prior research. We apply IC fingerprinting techniques by using power templates. Moreover, we use principal component analysis (PCA) and support vector machines (SVMs) to allow automatic Trojan detection. We show that these techniques are well-suited to reliably detect hardware Trojans on real ASIC designs.

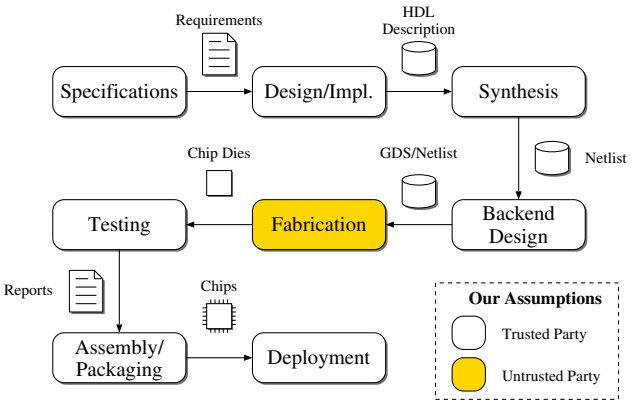

Figure 1: Design phases of today's ASIC development chains and their respective outputs.

Organization

The remainder of this paper is structured as follows. Section 2 gives a brief overview of the different design phases of a modern ASIC development flow, and describes our attacker model. The design being infiltrated as well as the actual hardware Trojan are described throughout Section 3. In Section 4, we present our measurement setup for the side-channel analysis (SCA) is given. During Section 5, we present our results regarding the Trojan detection before we conclude our work in Section 6.

2. ASIC DEVELOPMENT CHAIN

A simplified design flow for a modern ASIC is given in Figure 1. The first step of the design process is the specification, which determines the requirements for the design. This stage represents the earliest level in which a hardware Trojan can be inserted. Note that such an insertion requires the least amount of technical skill, and since the addition is visible throughout the entire design process, there are many opportunities to detect a hardware Trojan inserted at such an early stage. The design is usually described using a hardware description language (HDL) which is then synthesized into a netlist of standard logic gates. The next step is to generate the physical mask layout that will be used as a blueprint for fabrication of the IC. Finally, the actual fabrication of the IC takes place, which results in individual dies that are tested, packaged, and delivered to the customer.

An adversary may target any of the design stages outlined above. At each progressive level of the development flow

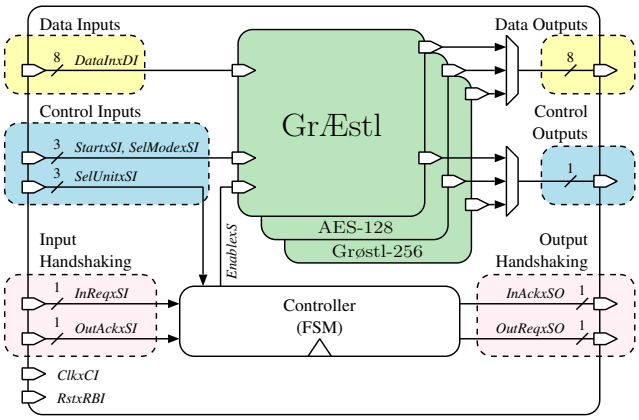

Figure 2: Chameleon top-level hierarchy.

the technical requirements continue to increase, and smaller companies usually need to outsource portions of the design flow to more specialized companies. Even for larger companies the actual IC manufacturing is outsourced to semiconductor foundries. Smaller companies may also need to outsource earlier levels of the design flow such as the *Back-end Design* or even earlier.

If not inserted during manufacturing, the last step at which a hardware Trojan can be inserted into the ASIC is just before the *fabrication* stage. If intercepted at this stage, very few technical details of the design will be known to the attacker, and she will have very little time to analyze the design and to insert the Trojan, as not to disrupt the tight manufacturing schedule. However, if inserted at this late stage, there will be virtually no opportunity to detect the Trojan before actually measuring the fabricated ICs.

Since there exist other concerns about the trustworthiness of the *Fabrication* design cycle related to integrated circuit piracy in general [3, 14], we believe that manufacturing is one of the most vulnerable stages in an ASIC's lifecycle.

2.1 The Attacker Model

For our real-world example, we assume the *Fabrication* to be the only untrusted stage in the development chain (cf. Figure 1). In this scenario, the attacker accesses the mask-layout data sent for fabrication, and adds the hardware Trojan. At this stage, the attacker has little information about the design itself, and due to the strict constraints of the manufacturing schedule would have little time to perform comprehensive reverse-engineering. In our example, we have also limited the amount of time the attacker has for the Trojan insertion to two working days. Furthermore, any mistake in the Trojan insertion could lead to a total failure of the overall design, which would most likely expose the attacker. As such, the adversary requires considerable technical skills to perform the attack. We show that it is possible to perform such an attack with reasonable assumptions.

Our attacker is a skilled designer, but has no a-priori knowledge about the design she wishes to infiltrate. The input/output cells used in any design are easily identifiable to anyone who has access to the mask-layout. Furthermore, common standard cells such as flip-flops can be detected with relative ease by an experienced layout engineer. This also leads to the identification of critical nets such as clock and reset in the design. The attacker used this information to evaluate different locations where to insert the Trojan and did not make use of any other design-specific information. Note that, to a large extent the Trojan circuit itself can be designed well in advance; only the small interface (a couple of inputs and outputs) needs to be attached to the actual circuit for a successful insertion. The space for the Trojan is obtained by replacing *filler cells* inserted into the design where normally no standard cells can be placed.

Table 1: Chameleon key properties.

Property		
Supply voltage (core/pad)	1.8/3.3	V
Core area	38,000	GE
Maximum operating frequency (f_{max})	125	MHz
Latencies (incl. I/O interface)		
AES encryption (128-bit plaintext)[1]	945	cycles
AES decryption (128-bit ciphertext)[1]	1,558	cycles
Grøstl hashing (512-bit message)[1]	3,465	cycles

[1] Applies for both standalone and GrÆStl version.

3. TARGET CIRCUIT

For our investigations, we could have chosen any existing ASIC design with a well-defined interface, regardless of its functionality. We had access to a number of student designs that were due to be submitted for fabrication. Our victim circuit Chameleon was designed independently from this work and includes three cryptographic cores[1]. The first core is a standalone implementation of the Advanced Encryption Standard (AES) [11], and the second core implements the cryptographic hash function Grøstl [5], which was one of the finalists of the SHA-3 hash competition [10] organized by the National Institute of Standards and Technology (NIST). Finally, the third design hosted on Chameleon is called GrÆstl and combines both AES and Grøstl in a single optimized datapath. Note that, all three cores are completely independent of each other, and the chip is configured to run one of the cores at a given time. The key properties of the Chameleon implementation are given in Table 1.

As can be seen from Figure 2, Chameleon communicates with its environment based on an eight bit wide I/O interface. A four-phase handshaking [12] has been implemented to control the data flow. The signals *InReqxSI, InAckxSO* and *OutReqxSO, OutAckxSI* are used for the input and output handshaking, respectively. For the Trojan insertion, the *red team*, which did not include the designer of Chameleon, used no design specific information. However, it made use of the four-phase handshaking signals as will be discussed in the following subsection. Strictly speaking, this information would not necessarily be directly available through a mask-layout level data transfer that we assumed in our model. Yet we believe that it is not difficult to extract this infor-

[1]Choosing a cryptographic design has made some of the choices in the project simpler. However, our approach is generic enough, and we are confident that we would have achieved similar results with any of the other designs as well.

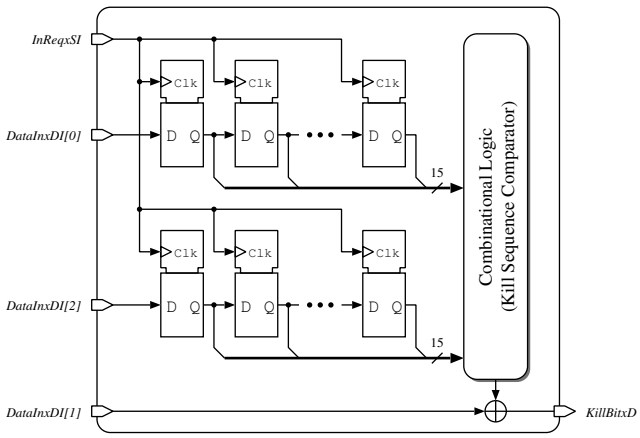

Figure 3: The DoS hardware Trojan.

mation, either through cursory examination of the circuit or through side channels such as pin names or supporting documentation.

3.1 The Trojan

The goal of the *red team* was to design a Trojan that was difficult to detect. This called for an extremely small circuit as shown in Figure 3. The Trojan observes the input of the genuine circuit and waits until a specific "kill-sequence" is detected. At this point, one other input-bit of the genuine circuit is flipped, which causes functional failure in the following clock cycles resulting in a denial-of-service attack. Unlike most published hardware Trojans, this design is continuously active, and is placed in a high-activity region of the ASIC. Since the "kill-sequence" is observed through a serial interface, there is a trade-off between the size of the circuit, and the length of the "kill-sequence". For this design a 30-bit sequence[2] was used, i.e., the same number of flip-flops were needed, which make up the majority of the area of the hardware Trojan. The Trojan has been designed to tap into two input bits in parallel resulting in the architecture shown in Figure 3. The additional connections can be placed close to the original connections, and add very little additional load to the genuine circuit. The current sequence is continuously compared to the "kill sequence" using a simple combinational circuit. Once the sequence is detected, the result is used to XOR one of the other data inputs. Once again, there is very little intrusion to the genuine circuit; only a comparatively small logic gate is added into the signal path, which (since it is on an external I/O path) is in most designs unlikely to be a timing critical path.

One remaining problem with regard to the Trojan insertion is the clock connection needed for the flip-flops used in the design. The *red team* chose not to use the regular clock signal, but derived it from the *InReqxSI* signal of the genuine circuit which is used to identify a new valid data item. Additional buffers are inserted in the clock path of the Trojan to reduce the load on the *InReqxSI* signal.

[2]In our case, we knew that the chip would not undergo exhaustive functional tests after production, so we were able to use a slightly shorter sequence to keep the Trojan area small. In a more realistic scenario, our victim circuit would also be more complex, allowing us to hide a longer "kill sequence" that would withstand even a very thorough functional test with a higher probability.

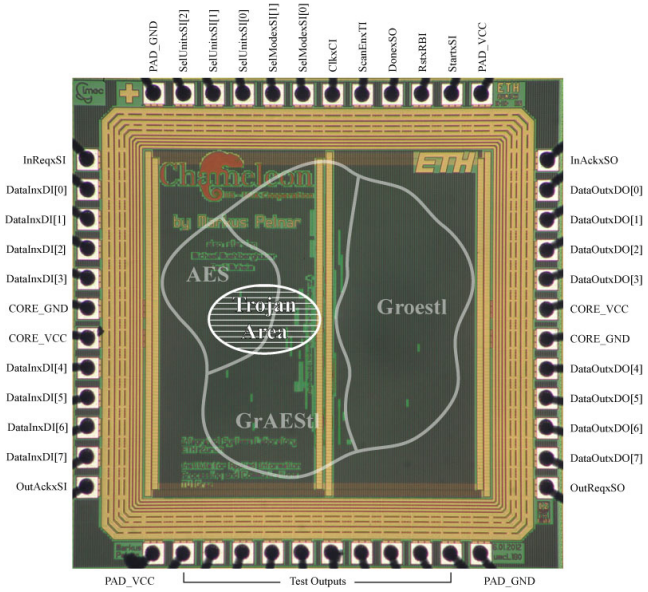

Figure 4: Chip photo of Chipit incl. pinout.

The actual insertion of the Trojan was performed based on a three-step approach: First, we identified the I/O pads and followed their connections. We needed to find four I/O signals: one for the clock, two for the kill sequence observation, and one that will be flipped by the kill-switch. Once some candidate I/O signals had been identified, a suitable region on the ASIC was searched. The goal was to find an area that was close to the connections, and had sufficient room to place the Trojan. The approximate area, where all the cells belonging to the Trojan are located, can be seen in Figure 4. In the second step, we removed some of the filler cells of the Chameleon design in order to place the standard cells required for the Trojan circuit. Finally, the signal and clock routing was completed using the available gaps in the actual circuit. We removed the original connection of input *InxDI[1]*, i.e., the "kill bit", and connected the combinational logic as well as the shift registers of the Trojan by adhering to the design rules of the target technology. All these steps were accomplished only on the mask-layout level data in GDSII format. The data was read into Cadence Design Systems Virtuoso 5.1.41 (isr20110503) EDA design software. All changes were done at the mask level, and the final layout was exported in GDSII format again. Both Chameleon and its malicious counterpart Chipit, were then fabricated using the 180 nm CMOS process by UMC. Figure 4 shows a chip photo of Chipit with the different components being highlighted. Due to the integration of the 30 flip-flops, the resulting Trojan requires a "comparatively large" area, i.e., about 190 GE, which is equal to 0.5 % of the overall Chipit design.

Note that, in a typical scenario, it would be possible to devise a library of Trojan sub-cells, such as different sequence detection and payload activation circuits, well in advance of the actual insertion. The attacker could then investigate the victim circuit and choose a suitable Trojan architecture from this pre-designed library without spending any time on the Trojan design during the actual insertion process.

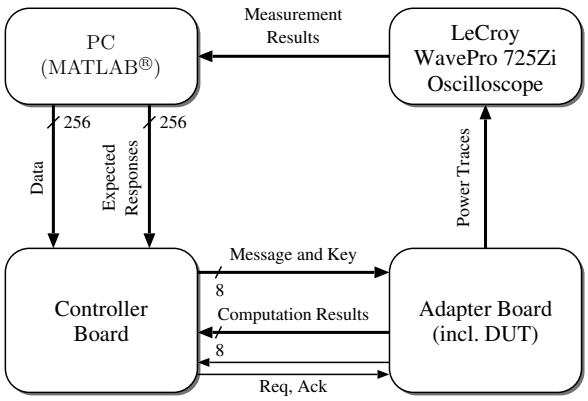

Figure 5: Schematic view of our measurement setup.

Several different hardware Trojan taxonomies exist in the literature (see, for instance, [15, 2, 7]). If we take these into account, we can classify our Trojan as follows:

Design phase:	Fabrication
Activation:	Externally triggered
Effect:	Denial-of-service
Location:	I/O
Abstraction level:	Gate level

4. THE MEASUREMENT SETUP

In total, around sixty chips of both Chameleon and Chipit were manufactured. Out of these a mix of eight chips were bonded in a QFN56 package. The final bonded group contained three Chipit and five Chameleon chips as listed in Table 2. These were then sent to the *blue team* for Trojan detection. The *blue team* was given the datasheet for Chameleon, and was able to design a test setup to correctly run the chip. The only a-priori information for the *blue team* was the pin layout of the packaged ASICs. No additional information, not even details of the mix were relayed from *red team* to *blue team* until the end of the study. The eight chips were tested as described in the following sections.

Our measurement setup mainly consists of four parts: an adapter board with a socket for our (Trojan infected and non-infected) ASICs, a controller board, a digital oscilloscope, and a personal computer (PC). In order to perform side-channel analyses, we decided to design a printed circuit board (PCB) that allows flexible power measurements of all available ASICs using an open-top QFN56 socket. This PCB is connected to a controller board (thus further on called *adapter board*). The controller board features an FPGA that is needed to provide all necessary interface signals to Chameleon/Chipit. The controller board is then connected

Table 2: Distribution of chips for measurement.

Chip	Design	Chip	Design
#1	Chipit	#5	Chameleon
#2	Chameleon	#6	Chameleon
#3	Chameleon	#7	Chipit
#4	Chipit	#8	Chameleon

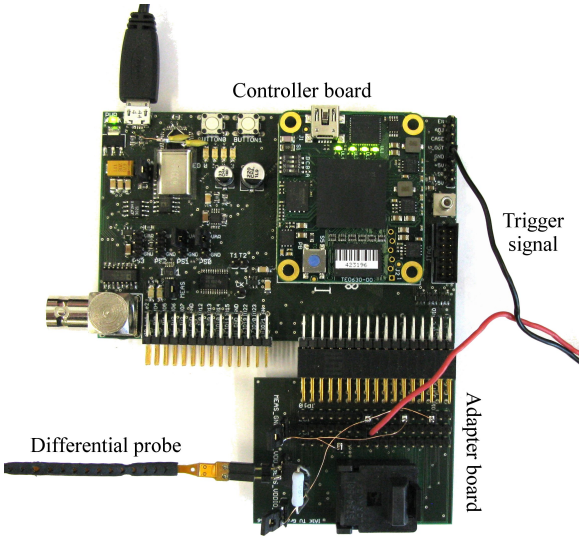

Figure 6: The controller board and the QFN56 adapter board.

to a personal computer that controls the overall measurement process. We use MATLAB® for this purpose which also offers sophisticated analysis tools to successfully perform side-channel analyses. Figure 5 shows a schematic view of our setup. In the following, we describe the used components and connections in more detail.

The adapter board has 51 pins that connect the Device Under Test (DUT) with the controller board. Only 48 pins are used by the DUTs; the remaining pins are still connected but not used in our experiments. The main component of the adapter board is the QFN56 socket that allows an easy exchange of the individual DUTs. In order to measure the power consumption, the board provides three additional pin headers: one ground, one VDD_IO, and one VDD_core pin header. Therefore, the voltage drop across a measurement resistor can be measured with the oscilloscope in the ground or power lines of the I/O and core of the DUT. In our experiments, we measured the voltage drop of a one Ohm shunt in the VDD_core line because it contains less noise from the ground or I/O communication. In addition to the power-measurement pins, all I/O pins of the DUT are available as pin headers to facilitate triggering on different I/O signals, e.g., on the start signal of the AES implementation (*StartxSI*).

The controller board and the adapter board are shown in Figure 6. The controller board is connected to the PC using a serial-over-USB interface. Core component is a Xilinx Spartan-6 FPGA that is fully flexible in terms of pin configuration and control. It mainly implements a serial interface (UART) to communicate with the PC and the protocol handling to communicate with the DUT (see Figure 7 for a schematic view of the top module). Both modules are connected via an AMBA APB bus (*Serial2APB*). In order to communicate with the DUT, we implemented the following protocol. First, we select the proper mode of operation of the DUT. We decided to target the stand-alone version of AES ($SelUnitxSI = 1$) and perform encryptions only ($SelModexSI = 0$). Second, the PC sends a 128-bit AES key and a 128-bit message to the controller board. For

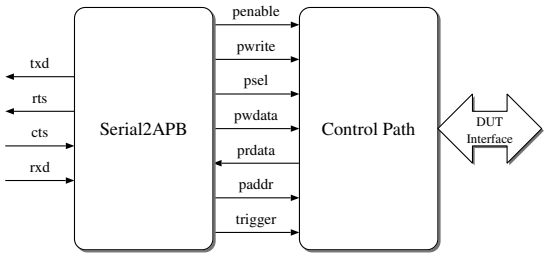

Figure 7: Schematic view of the controller FPGA top module.

this, we decided to use an AMBA data width of 256 bits to send 256 bits within one write cycle. This significantly improves the measurement performance. The data is then stored in a 256-bit register and split into 8-byte chunks to interface with the DUT. After AES encryption, the data is stored into a register and read out using the AMBA APB read command. The ciphertext is then compared with an AES software implementation output using MATLAB®.

The power consumption is measured using a LeCroy WavePro 725Zi oscilloscope. We used a sampling rate of 1 GS/s and captured the entire AES encryption operation (including I/O communication). As a differential probe we used an active probe of type LeCroy Dx20-SP. The oscilloscope is connected to the PC over LAN and transmits the acquired power traces in a sequence mode, i.e., 100 AES encryptions are executed before sending the power trace blocks over LAN.

For the following power analyses, we set the baud rate to 19,200 bps and the DUT clock frequency to 10 MHz. Furthermore, we kept the AES key and the input constant in order to reduce noise that is caused by the data. For each DUT, we measured one million power traces which took about 2.5 hours per DUT. This amount of traces is required because 1) we expect a very low signal from the Trojan hardware because of its small area footprint, 2) the Trojan is always active and never in a *dormant* state which makes it hard to detect because of missing activation signals, and 3) the more traces are used for averaging, the more noise is reduced which helps us to characterize the Trojan signals.

5. ANALYSIS RESULTS

After measuring a set of one million power traces per DUT, we performed the following analyses. First, we built templates of the measured power traces for each ASIC and applied Differential Power Analysis (DPA) techniques to identify Trojan-dependent signals. Second, we used principal component analysis (PCA) to allow hardware-Trojan detection and classification. Finally, we tried to improve the results using support vector machines (SVMs) and trained them with test data to automatically allow a classification with a high success rate, i.e., 100 %.

5.1 Building Templates

In a first observation, we wanted to answer the question if we can identify major differences and high variances between the power traces of Chameleon and Chipit. For this purpose, we built power templates by calculating the mean of all power traces from each ASIC. After that, we calculated the difference of means (DoM) and took the absolute

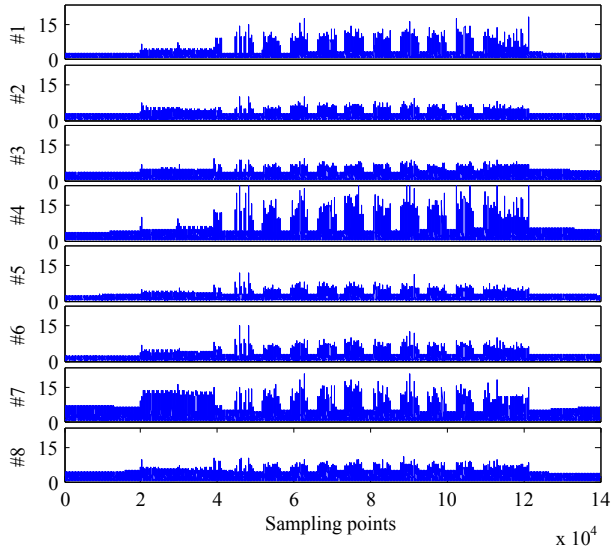

Figure 8: Difference of means of all DUTs.

value of the result for a better visualization. Figure 8 shows the result. From the DoM and also the plain power traces, we can identify the following operations: the loading of the input data (I/O communication) occurs between the sampling points $20,000-40,000$. After that, an AES encryption is performed (we can clearly discern the ten rounds of AES between $40,000-110,000$ sampling points), and the ciphertext is transmitted back to the controller board at sample points $110,000-120,000$. Having a closer look at the differences, it shows that chip #1, chip #4, and chip #7 have a higher variance compared to the remaining chips but the differences are not significant using DoM. Furthermore, it shows that there are no distinct sample points that provide high differences due to Trojan activity. Instead, it shows that these differences are widespread over the entire power traces which corresponds to our expectations since the injected Trojans are always active and never dormant. In the following, we zoomed into regions where the variance is high to characterize the higher variance and to identify Trojan-dependent signals.

Figure 9 shows 20 sampling points of all DUTs during a region where the differences between the datasets are high. We marked the sampling points of chip #1, chip #4, and chip #7 in different colors (blue, red, and green); the remaining points are printed in gray. It shows that the mean power of Chipit differs from the mean power of Chameleon in several points. At point 5, chip #1 and chip #4 provide a higher mean power while chip #7 has a lower mean power compared to the rest. At point 6, it is the other way around. We could identify such differences in many sampling points throughout the mean power traces (usually at strong peaks that are caused, for instance, by the internal AES clock signal). For all differences, however, we could observe that the mean power of chip #1, chip #4, and chip #7 differs significantly from the other mean power traces, e.g., chip #1 and chip #7 might have a higher mean power than chip #4, as opposed to our given example, or all chips have higher or lower mean power etc. So the mean power of Chipit and Chameleon significantly differs at some points both in either amplitude and/or time dimension.

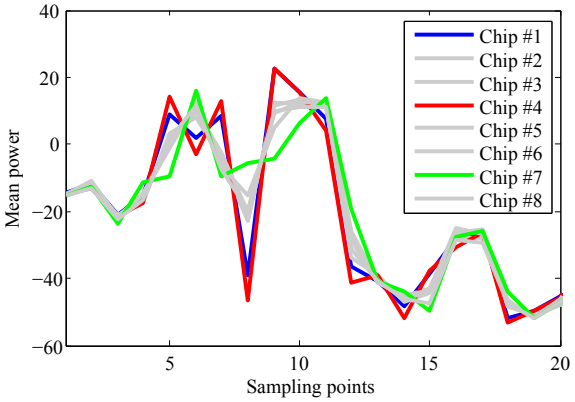

Figure 9: Sampling points of mean power traces.

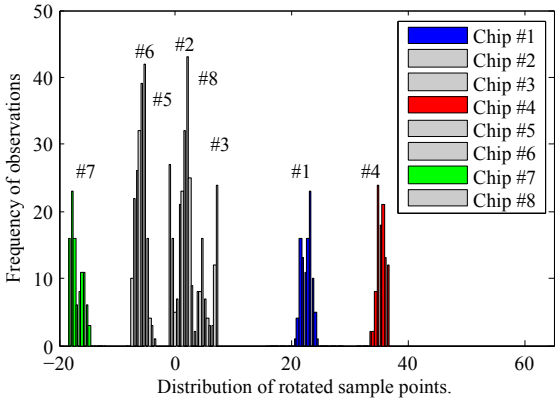

Figure 10: Result of a PCA-based classification.

5.2 Principal Component Analysis

In order to improve our results, we apply principal component analysis (PCA) in order to reduce noise and to efficiently discover Trojan-dependent signals. PCA, in that context, has been first proposed by Agrawal et al. [1] in 2007. They proposed to apply the Karhunen-Loève (KL) method to allow integrated circuit (IC) fingerprinting and detection of hardware Trojans. They evaluated the method for simulated power traces and marked the analysis of "real fabricated ICs" as future work.

For each ASIC, we calculated 100 mean power traces where 10,000 traces have been used for averaging. From these mean traces, we chose 20 sample points (further called *points of interest*) which have a high variance or difference from all other means. In fact, we chose the same points as shown in Figure 9. Thus, we obtained a matrix of mean power traces with 800×20 elements. After that, we transformed this higher-dimensional dataset into principal components by finding linear combinations of the original sample points that provide less redundant information. This is done by first calculating the covariance of all points of interest. Second, we calculated the eigenvectors resulting in 20 different principal components. After sorting the eigenvectors, we chose the first principal component which provides the highest variance and therefore most interesting information. The feature vector is then multiplied with the original traces to obtain the rotated (transformed) traces. Figure 10 illustrates the result of the analysis in form of a histogram. It shows the distribution of the sample points of each rotated trace. The probability distributions of chip #1, chip #4, and chip #7 have a high variance compared to all other chips. They can be therefore easily identified and classified into ASICs with a Trojan and chips without a Trojan.

5.3 Support Vector Machines

We also decided to evaluate support vector machines (SVM) to allow an automatic Trojan detection. SVMs can be basically used to classify a set of power traces to a hardware Trojan (Chipit) or non-Trojan (Chameleon) group after training them with sample data. If trained successfully, they allow an efficient detection of manufactured ICs that do not map to the trained (high-dimensional) feature space of non-infected chips. Hospodar et al. [6], for example, successfully applied this technique in the context of side-channel analysis.

We used the statistics toolbox in MATLAB® to perform the SVM classification. With the built-in method `svmtrain` the SVM can be trained. As an input, `svmtrain` takes the training data containing the feature values of the measured power traces. Similarly to PCA, this training data is represented by a matrix with R_{td} rows and C_{td} columns where R_{td} equals to the number of training data and C_{td} equals to the number of features. Second, a group vector with length R_{td} is required. This vector defines the group membership of the training data. A third parameter defines the used kernel function for the SVM. For the classification in our experiments, we applied a *quadratic* kernel function which provided the best performance. The method `svmclassify` is used in order to classify new data according to the previously trained SVM. `svmclassify` requires only one input, the feature data which has to be classified. The classification result is returned by the method `svmclassify`.

Figure 11 shows the result of one classification of the chips using the SVM. The feature vector for this classification is two-dimensional. In order to find suitable points for the feature vector, the variance values for each time instance of the power traces are considered. As equally done for PCA, we decided to consider all sample points with a high variance as points of interest. Therefore, we chose the same points of interest as used for PCA to allow a comparison of the results. In total, 800 traces were available for the experiments which equals to 100 traces per chip. Ten feature vectors from chip #1, chip #2, and chip #7 served as training data for the SVM. It shows that 30 out of 800 traces (equals 3.75%) are sufficient in order to correctly classify the remaining data. A further experiment showed that one trace per chip (equals eight traces in total) for training is also sufficient in order to perform a classification without errors. The resulting classification clusters are illustrated in Figure 11. With this approach, the Chipit chips (chip #1, chip #4, and chip #7) can be automatically distinguished from the Chameleon chips (chip #2, chip #3, chip #5, chip #6, and chip #8). This experiment assumes that for training the SVM chips without and with a hardware Trojan are available. If, for example, only chips without a hardware Trojan are available for training, some sort of threshold detection based on the feature vector can be used in order to classify new chips according to their membership.

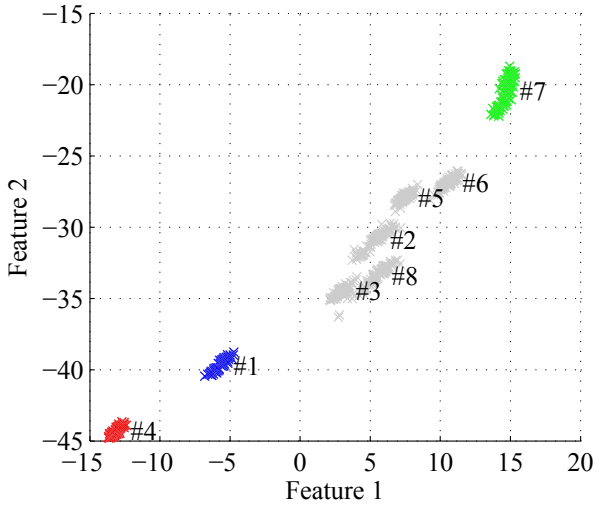

Figure 11: Result of a SVM-based classification using a two-dimensional feature vector.

6. CONCLUSION

In this work we have demonstrated that it is indeed possible to design a small, effective hardware Trojan with little or no previous knowledge about the inner functionality of the victim circuit. The Trojan was inserted at the mask-layout level, just prior to the actual fabrication stage using only physical layout modification. Even though, such changes are considered to be very difficult, we have demonstrated that it can be accomplished with relative ease, in our case, in less than two working days.

We have manufactured two sets of ASICs, one with and the other one without the aforementioned hardware Trojan. This represents the first time, such an effort has been published. A set of manufactured chips were sent for analysis to a different group which was given no information on the Trojan hardware at all. All three chips containing Trojans were correctly identified out of a mixed population of eight samples using power-based side-channel analysis. A principal component analysis based approach was used to refine the power analysis results and to generate clearly identifiable distributions. In addition, an automated classification approach using support vector machines was also demonstrated. Our results clearly show that it is possible to reliably distinguish chips containing even a very small Trojan (less than 0.5 % of the total area) by using power based side-channel analysis.

As future work, we would like to incorporate not only a subset of chosen sampling points (points of interest) but all measured data points. This is interesting for following an automatic Trojan detection approach. Furthermore, we plan to characterize the Trojan and noise signals in more detail to answer open questions such as "at which regions in time is the Trojan signal higher or lower than the process noise?" or "what is the exact reason for the higher or lower mean power consumption of chip #1, chip #4, and chip #7?". Finally, we plan to make a detailed boundary analysis for our classification techniques to evaluate the lower bound for a 100 % success rate.

7. REFERENCES

[1] 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*, pages 296–310, 2007.

[2] R. Chakraborty, S. Narasimhan, and S. Bhunia. Hardware trojan: Threats and emerging solutions. In *HLDVT'09*, pages 166–171, Nov. 2009.

[3] CRN. Cisco Channel At Center of FBI Raid On Counterfeit Gear. Webpage (Last accessed on 2013-04-28), May 2008. http://www.crn.com/news/networking/207602683/cisco-channel-at-center-of-fbi-raid-on-counterfeit-gear.htm.

[4] Defense Science Board. Defense Science Board Task Force On High Performance Microchip Supply. Technical report, US DoD, Feb. 2005. http://www.acq.osd.mil/dsb/reports/ADA435563.pdf.

[5] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (Round 3), Mar. 2011. http://www.groestl.info/Groestl.pdf.

[6] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302, 2011.

[7] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *IEEE Computer*, 43(10):39–46, Oct. 2010.

[8] C. Lamech and J. Plusquellic. Trojan Detection based on Delay Variations Measured using a High-Precision, Low-Overhead Embedded Test Structure. In *HOST'12*, pages 75–82, 2012.

[9] C. Lamech, R. Rad, M. Tehranipoor, and J. Plusquellic. An Experimental Analysis of Power and Delay Signal-to-Noise Requirements for Detecting Trojans and Methods for Achieving the Required Detection Sensitivities. *Information Forensics and Security, IEEE Transactions on*, 6(3):1170–1179, 2011.

[10] NIST. SHA-3 Cryptographic Hash Algorithm Competition. Webpage (Last accessed on 2013-04-26). http://csrc.nist.gov/groups/ST/hash/sha-3/.

[11] NIST. *Advanced Encryption Standard (AES) (FIPS PUB 197)*. National Institute of Standards and Technology, Nov. 2001.

[12] A. M. G. Peeters. *Single-Rail Handshake Circuits*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1996.

[13] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran, and K. Rosenfeld. Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges. *Computer*, 44(7):66–74, 2011.

[14] E. Times. Fake NEC company found, says report. Webpage (Last accessed on 2013-04-28), May 2006. http://eetimes.com/electronics-news/4060352/Fake-NEC-company-found-says-report.

[15] X. Wang, M. Tehranipoor, and J. Plusquellic. Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions. In *HOST'08*, pages 15–19, June 2008.