Advanced Techniques for Designing Stealthy Hardware Trojans *

Nektarios Georgios Tsoutsos
NYU Polytechnic School of Engineering
New York City, USA
nektarios.tsoutsos@nyu.edu

Charalambos Konstantinou
NYU Polytechnic School of Engineering
New York City, USA
ckonstantinou@nyu.edu

Michail Maniatakos
New York University Abu Dhabi
Abu Dhabi, UAE
michail.maniatakos@nyu.edu

ABSTRACT

The necessity of detecting malicious modifications in hardware designs has led to the development of various detection tools. Trojan detection approaches aim to reveal compromised designs using several methods such as static code analysis, side-channel dynamic signal analysis, design for testing, verification, and monitoring architectures etc. This paper demonstrates new approaches for circumventing some of the latest Trojan detection techniques. We introduce and implement stealthy Trojans designs that do not violate the functional specifications of the corresponding original models. The designs chosen to demonstrate the effectiveness of our techniques correspond to encryption algorithms and a pseudo random number generator. The proposed Trojans are inserted into the original RTL, and decrease the overall security of the designs, minimizing detection probability by state-of-the-art static analysis tools.

Categories and Subject Descriptors

B.6.2 [**Hardware**]: Logic Design—*Security and Trust*

Keywords

hardware, security, trojans, backdoors, intellectual property

1. INTRODUCTION

The topic of Integrated Circuit (IC) security is becoming important due to the fact that ICs are embedded in nearly every piece of modern electronic equipment. The contemporary globalized economy of the semiconductors design has increased the rate of malicious activities, therefore System-on-Chips (SoCs) designs are more vulnerable than any time in the past. In addition, the use of Intellectual Property (IP) components from potentially untrusted design houses, only makes the problem worse.

Hardware Trojans consist of malicious modifications to the design of ICs and typically induce unwanted effects [1, 2]: Trojans can leak secret information, downgrade performance, change the functionality of the target device and also can be used for Denial-of-Service (DoS) attacks. Trojans can be mounted into ICs at multiple stages of design and manufacturing process, and may even have a zero hardware overhead [3]. In this paper, we concentrate on hardware Trojans inserted at the Register Transfer Level (RTL), and we propose new techniques for designing stealthy Trojans while minimizing detection probability by modern detection techniques.

The RTL design is very flexible to implement any malicious function since the attacker can potentially have full control of the generated hardware. Furthermore, since an RTL modification affects all taped-out chips, a golden model cannot be used for power and timing dynamic analysis to expose the Trojans; in addition, process variation may also conceal carefully crafted Trojans of very low overhead. Functional analysis can also be very useful for Trojan detection at the RTL level, but a malicious designer may still minimize detection rates using seemingly random functional pattern sequences. Given the complexity of the problem, there is no panacea for detecting every possible malicious function.

In this work, we assume the role of a malicious insider within a design house that sells IP to other organizations. In our assumed threat scenario, the organizations receiving the malicious IP may apply only static analysis methods on the Hardware Description Language (HDL) code of the IP, without actually simulating or implementing the design (dynamic analysis). We primarily focus on circumventing state-of-the-art static analysis detection techniques, such as [4], while ensuring there is no modification in the functional specification of the target designs (or ensuring that such modifications are undetectable without exhaustive exploration of all system states).

The proposed detection evasion techniques can be summarized as follows:

1. modifying the control logic of the target design, so that multiple levels of nested finite state machines (FSMs) are used instead of a single level, and as soon as a trigger condition is fulfilled, a malicious nested FSM transition is executed.

*The hardware trojan designs presented in this paper have received the first place award in the sixth Embedded Systems Competition at the 2013 Cyber Security Awareness Week.

DAC '14 San Francisco, California USA

2. exploiting the negative edge of the clock in a synchronous design, as well as infrequently used inputs of the IC, to modify the internal states on the design in a stealthy fashion.

These techniques are demonstrated, without loss of generality, using three sample Trojans in encryption hardware accelerators for the Data Encryption Standard (DES) algorithm, the eXtended Tiny Encryption Algorithm (XTEA) and a Pseudo Random Number Generator (PRNG) based on one-dimension Cellular Automata (CA). These accelerators are designed to be very simple and have low area overhead, so additional IC testing hardware (such as scan chains) is missing, and standard testing techniques cannot be applied (i.e. verification would require exhaustive state exploration).

2. CRYPTANALYSIS: ROUND REDUCTION ATTACKS AND ATTACKS ON PRNGS

2.1 Block ciphers properties

The use of block ciphers advocates in constructing secure encryption schemes. A block cipher is a function $E : (0,1)^k \times (0,1)^n \rightarrow (0,1)^n$. Function E takes a key of k-bits and a plaintext of n-bits and results an n-bit ciphertext. Block ciphers themselves are not secure encryption mechanisms but are the necessary building blocks that will provide security to a model.

According to Claude Shannon, the basic techniques for obscuring the redundancies in a plaintext are *confusion* and *diffusion* [5]. Confusion obscures the dependency between the ciphertext and the key and in substitution ciphers confusion is typically added to the encryption process. On the other hand, diffusion refers to the dissipation of the redundancy of the plaintext by spreading it out over the ciphertext; permutation ciphers are typically designed to cause diffusion.

2.2 Attacks on block ciphers

Secret key cryptographic algorithms such as DES and XTEA are based on a round function that is computed iteratively to ensure the security of the algorithm against attacks. In general, most modern block cipher systems apply a number of rounds in succession to encrypt or to decrypt a message. The number of rounds is important to the strength of ciphers. In case the number of rounds executed is less that the number of the necessary rounds, the plaintext is not sufficiently shuffled, and it is possible to ultimately break the encryption (differential cryptanalysis).

In Shannon's terminology, more rounds lead to greater confusion and diffusion [5]. Reduced rounds in a cryptographic algorithm typically result into Conditionally Computationally Secure (CCS) ciphers [6]. Most attacks on block ciphers begin by attacking versions with reduced number of rounds: for example, in the DES algorithm, after 7 rounds the right half of the output is affected and after 8 rounds the left half is affected. The normal operation of DES with 16 rounds, however, takes advantage of an *avalanche effect* ensuring that similar inputs yield completely different and independent looking outputs.

2.3 Attacks on PRNGs

Pseudo Random Number Generators (PRNG) are constructions designed to produce values that (in polynomial time) cannot be distinguished from truly random values of the same length. This is the most important property of PRNGs and any modification or attack that allows to control or degrade the randomness of the output is substantial. In general, attacks to PRNG essentially relate to distinguishability of output from truly random streams, setting the PRNG to a chosen internal state, predicting the future outputs of the PRNG as well as reducing the number of possible different output values.

Attacks on PRNGs are classified into three main classes [7]: *Direct Cryptanalytic Attack* when the attacker is able to distinguish between random and PRNG outputs; *Input-Based Attacks* when the attacker is able to distinguish between random and PRNG outputs by using knowledge or control of the PRNG inputs; and *State Compromise Extension Attacks* when the attacker is able to distinguish between random and PRNG outputs from before an internal state was compromised or recover outputs from after the PRNG has collected a sequence of input which the attacker cannot guess.

3. A HARDWARE TROJAN IN DES

3.1 General Background on DES

The Data Encryption Algorithm (DES) is the most studied symmetric-key block cipher. It was designed to encipher sensitive but non-classified data. DES is a bit-oriented algorithm and it is defined in US standard FIPS 46-2 [8]. The design of DES is also related to product ciphers and Feistel network ciphers.

DES accepts a 64-bit plaintext block and produces a 64-bit ciphertext block using a 56-bit key. The 64-bit input plaintext is divided into two halves, left and right, each consisting 32 bits. The splitting is done by rearranging the bits. A similar swapping of bits is implemented at the end of the encryption to create the 64-bit output ciphertext from the two halves. DES consists of 16 equivalent rounds, which means that the round function of DES is applied 16 times-sequentially to the two halves. Each round uses a separate *subkey* of 48 bits. The round subkey is formed by selecting 48-bits from the 56-bit key using left shift and the permutation tables $PC-1$ and $PC-2$ referred in [6]. In case the order of the selected keys of each round is reversed, then the input is deciphered.

3.2 Trojan Implementation

The proposed Trojan aims to decrease the overall security of the DES algorithm by attacking the key schedule (based on the round selection). The idea behind this Trojan implementation is to reduce the probability for detection tools to identify the malicious functionality. Specialized detection techniques focus on tracing Trojans based on rare bit upsets. Our proposed Trojan, however, employs commonly used logic. The sequential character of the designed Trojan is implemented using four nested FSMs (instead of a simple state counter) that are always active and hold the current round number of DES. The Trojan is ultimately triggered when certain sequences of internal signal conditions and states appear.

3.2.1 Trigger Mechanism

Without loss of generality, the DES Trojan is triggered when chosen patterns are applied for encryption or decryp-

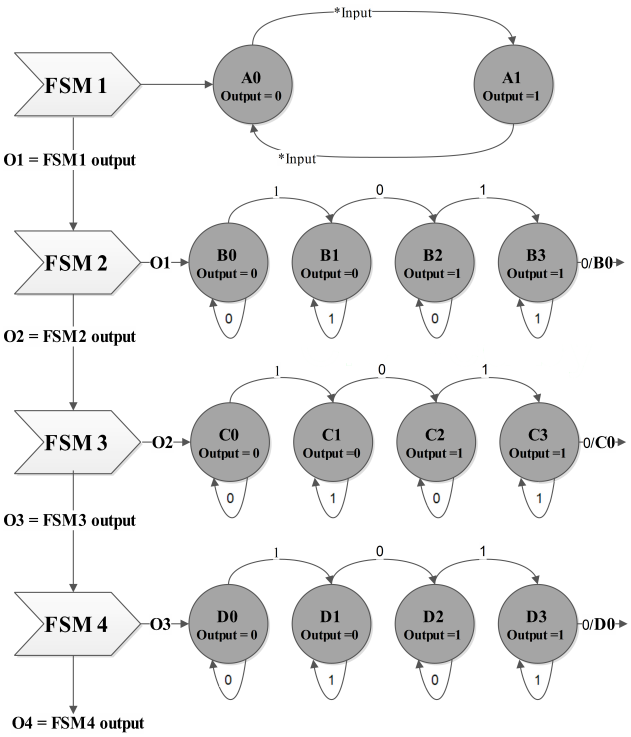

Figure 1: Modified DES FSM State Diagram.

tion to the DES block. The states of the four nested FSMs correspond to one of the 16 rounds in the DES algorithm. *Therefore, detection techniques that are based on rarely upset bits are expected to fail identifying the Trojan, since its activation mechanism depends on frequently-used states inside the nested FSMs.* The FSM model implemented for this proposal is illustrated in Fig.1. For simplicity, in our design we chose to activate the Trojan if and only if specific input patterns containing 6 or less binary 1s at any place, are entered. The probability for the malicious modification to be activated is thus once every $2 * 10^{11}$, using random inputs (calculating the combinations of having at most 6 ones in a random 64 bit value). Detecting this modification during verification, however, would require exhaustive search of at least $2 * 10^{11}$ inputs.

3.2.2 Payload

The inserted Trojan aims to reduce the rounds of DES and consequently produce identical subkeys for a number of rounds. When the trigger mechanism is activated, the round number (determined by the 4 nested FSMs) repeats *only 4 out of the 16 rounds*. Therefore, our malicious modification compromises the original circuit's normal operation without being suspicious for detection tools.

4. A HARDWARE TROJAN IN XTEA

4.1 General Background on XTEA

The eXtended Tiny Encryption Algorithm is the result of the efforts of cipher designers David Wheeler and Roger Needham [9] and consists of a 64-bit block *Feistel network*. The algorithm uses a large key (128 bits) and a secure implementation requires 32 iterations of 2 Feistel rounds each.

Based on the best available cryptanalysis today, XTEA is vulnerable to related-key differential attacks on 27 out of the 64 total Feistel rounds, as suggested in [10]. Thus any implementation of XTEA that iterates less that 13 times (i.e. 26 Feistel rounds since each iteration has 2 such rounds), is considered insecure.

4.2 Trojan Implementation

The proposed Trojan is designed to decrease the security of XTEA encryption operation by attacking the *round scheduling* of the algorithm. Even though specialized detection techniques allow detection of rare upsets in the circuit, this Trojan is dependent on the number of iterations of states in the internal round scheduling state machine. *All states in the sequence are being frequently used, and thus detection techniques based on rare events are expected to fail detecting such Trojans.* Specifically, the proposed Trojan degrades the security of XTEA, since the effective number of rounds is less that the maximum number of rounds vulnerable to known cryptanalysis.

4.2.1 Trigger Mechanism

The trigger mechanism of this Trojan can be controlled by the user input, without loss of generality. The Trojan inspects the inputs to the design for a specific pattern, and based on this property releases the malicious payload. XTEA requires a plaintext input of 64 bits in total (block size). The input block is provided split into two halves of 32 bits each. The trigger mechanism inspects only one of these two halves and counts the number of bits in that half that are equal to the logic value '1' (the Trojan essentially counts all the 1's in the 32-bit wide half of the input). Then, based on a parameterized threshold, the Trojan determines if the number of ones in the sum is greater than that given threshold. In this case the Trojan triggers the payload.

For our implementation, the Trojan is activated if the first half of the input holds 16 or more 'ones'. This means that for a random input of 64 bits, the Trojan is activated once every $1.3 * 10^{10}$ inputs, based on the sum of combinations of inputs that match the activation conditions in the first half of the input (e.g. $32!/16! * (32 - 16)!$ for 16 ones) multiplied by the input probability in the second half of the input (2^{-32}). Detecting this modification during verification would also require exhaustive search of at least $1.3 * 10^{10}$ inputs.

4.2.2 Payload

As soon as the Trojan is activated by the sample trigger mechanism described above, the payload modifies the number of rounds executed by XTEA. Altering the number of rounds is detrimental for the security of the algorithm and the confidentiality of the encrypted plaintext. Specifically, the payload progresses the round iteration three times faster that normal, and thus XTEA performs 3 times fewer iterations (i.e. 10 iterations instead of 32). Based on the previous analysis, anything below at least 13 is known to be insecure since this would correspond to 26 Feistel rounds in total. Furthermore, the XTEA design API assumes the output is available only when the *ready* signal is set, so even though the result is computed sooner, any IC relying on the *ready* signal will not detect this modification.

5. A HARDWARE TROJAN IN CA PRNG

5.1 General Background on CA PRNG

Cellular automata (CA) is a discrete mathematical model that consists of a list of cells which have a value of `0` or `1` and are all updated at a certain period based on a set of rules [11, 12]. A one-dimension CA is simply a series of bits, each one of which changes its value based on rules that depend on the current bit value and the value of the bits to the left and right of the current bit (i.e. the bit neighbors). There also exist two dimensional CAs, with notable example 'Conway's Game of Life'.

Using a popular set of rules (called 'Rule 30') for updating the bits in a one-dimension (i.e. linear) CA, a PRNG can be constructed that applies a mask to only 32 bits at the center of the linear CA (which generally is continuously increasing in width). Rule 30 replaces the *middle bit* of each pattern `111, 110, 101, 100, 011, 010, 001, 000` with bits `0, 0, 0, 1, 1, 1, 1, 0` respectively (these 8 bits correspond to number 30 in decimal, hence the name of the rule).

Our design is an implementation of a PRNG that can be initialized at any pattern (and also follow any set of rules but uses Rule 30 by default). In order for the PRNG to be secure, it should output unpredictable 32-bit numbers. The security of the generator, however, is significantly affected if an attacker can control the output from the input in a stealthy manner, since the randomness of the output is lost.

5.2 Trojan Implementation

For this PRNG design, our proposed Trojan allows an attacker to provided one bit at a time for a desired PRNG output. Essentially the malicious user employs unary encoding to send one bit at a time in order to build a desired output of the PRNG. As soon as the Trojan collects all 32 bits from an input signal in a stealthy manner, the attacker-selected value is returned in the output of the design, following a final input toggle by the attacker.

5.2.1 Trigger Mechanism

The Trojan is triggered by the malicious user who cleverly manipulates one input signal when the clock signal is 'low' (i.e. from the negative edge until the next positive edge). The input signal is normally used only when the clock signal is high and thus the attacker can encode additional malicious behavior to that input signal. Without loss of generality, we exploit an input signal that is normally used (i.e. the clock signal is high) to request the next random value; when the clock signal is low, however, the malicious user encodes (in unary) the bits of a desired output value. Finally, the user toggles a second signal (normally used to update the PRNG seed) during the negative edge of the clock, and the stealthy input is immediately sent to the output. Therefore, since the Trojan is hidden in the negative edge of the clock, verification techniques not checking for negative edge events are expected to fail detecting the malicious logic.

5.2.2 Payload

The payload of the Trojan can be activated as soon as the malicious user has sent all 32 bits of the desired output or sooner. At that time, the PRNG replaces the correct (i.e. pseudorandom) output with the chosen value by the user. As soon as this value is sent to the output and the user toggles the signal normally used for updating the PRNG seed (but during the negative edge of the clock), the PRNG resumes generating random values. The malicious user is also able to repeat this process an arbitrary number of times.

6. CONCLUSIONS

In this paper we presented two techniques for minimizing the success rate of modern Trojan detection methods, such as [4], by exploiting deep state machine nesting and negative clock side channels. Our proposed techniques have been elaborated into three sample encryption hardware accelerator designs (DES, XTEA and CA PRNG), that are small and simple by nature and do not support additional testing hardware such as scan chains. The proposed designs have also been evaluated during the 6th Embedded Systems Competition of CSAW 2013 and received the first place award in the competition.

7. REFERENCES

[1] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[2] M. Tehranipoor, H. Salmani, X. Zhang, W. Xiaoxiao, R. Karri, J. Rajendran, and K. Rosenfeld, "Trustworthy hardware: Trojan detection and design-for-trust challenges," *Computer*, vol. 44, no. 7, pp. 66–74, 2011.

[3] N. Tsoutsos and M. Maniatakos, "Fabrication attacks: Zero-overhead malicious modifications enabling modern microprocessor privilege escalation," *IEEE Transactions on Emerging Topics in Computing*, vol. DOI 10.1109/TETC.2013.2287186, 2013.

[4] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis," in *ACM Conference on Computer and Communications Security*, 2013.

[5] C. E. Shannon, "Communication theory of secrecy systems," *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[6] N. Jorstad and T. Landgrave, "Cryptographic algorithm metrics," in *20th National Information Systems Security Conference*, 1997.

[7] J. Kelseyi, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic Attacks on Pseudorandom Number Generators," in *Lecture Notes in Computer Science*, vol. 1372, pp. 168–188, Springer, 1998.

[8] NIST, "Data Encryption Standard, FIPS PUB 46-2, National Institute of Standards and Technology," 1977.

[9] R. M. Needham and D. J. Wheeler, "TEA extensions," 1997.

[10] Y. Ko, S. Hong, W. Lee, S. Lee, and J.-S. Kang, "Related key differential attacks on 27 rounds of XTEA and full-round GOST," in *Fast Software Encryption*, pp. 299–316, Springer, 2004.

[11] S. Wolfram, "Random sequence generation by cellular automata," *Advances in applied mathematics*, vol. 7, no. 2, pp. 123–169, 1986.

[12] M. Sipper and M. Tomassini, "Generating parallel random number generators by cellular programming," *International Journal of Modern Physics C*, vol. 7, pp. 181–190, 1996.