Hardware Trojans in TRNGs

Honorio Martin, Pedro Peris-Lopez, Juan E. Tapiador, Enrique San Millan, and Nicolas Sklavos

***Abstract*—Nowadays hardware Trojans have become an increasing security threat for commercial and military systems and applications. Their effects in devices devoted to security purposes, which generally make use of integrated circuits, suppose a challenging problem to face by the industry. These security devices often support a True (or pseudo) Random Number Generator (TRNG or PRNG) to generate keys or guarantee freshness of the computed cryptographic tokens. For this, TRNGs play a key role in security systems which should be free of Hardware Trojans. To the best of our knowledge, the inclusion of a Hardware Trojan in a RNG has not been explored yet. In this paper, we present a study about the integration of Hardware Trojans in TRNGs. In particular, the tests, commonly used to evaluate the randomness of data streams, have been employed to define the features demanded to the Hardware Trojan with the aim of being effective and undetectable whether its output is only considered.**

***Index Terms*—TRNGs, Hardware Trojans, Hardware Trojan Activation, Hardware Trojan Effects, On-line Test.**

I. Introduction

A Hardware Trojan is a form of malicious circuitry that damages the function or/and trustworthiness of an electronic system. The increasing threat of Hardware Trojans has been reflected in the increasing number of publications related to this topic [1][2][3]. Hardware Trojans suppose a potential danger that is growing for the integrated circuits (ICs) industry. One of the main concerns is the use of Hardware Trojans in ICs oriented to security purposes, which are critical for many applications, and its malfunction would be catastrophic.

These ICs devoted to security usually embed a True-Random Number Generator (TRNG) that is in charge of generating fresh cryptographic tokens. As TRNGs play a key role in cryptographic systems, they typically include on-line testing capabilities to guarantee an uniform distribution of 0s and 1s at the output. ENT [4], DIEHARD [5], NIST [6] and AIS31 [7] battery tests are well-known suites used to evaluate randomness quality of a TRNG output. These mentioned on-line tests could hamper the integration of Hardware Trojans on TRNGs in security chips because any modification of the output might be detected by these tests. If the insertion of the Hardware Trojan is successful, the security block would be malfunctioning during certain time periods. Apart of this it has been show how Hardware Trojans also imply risks on economy and society [8].

H. Martin and E. San Millan are with Department of Electronics Technology, Universidad Carlos III de Madrid, 28911 Leganés, Madrid, Spain. (E-mail: {hmartin, quique}@ing.uc3m.es)

P. Peris-Lopez and J.E. Tapiador are with the Computer Security (COSEC) Lab at the Department of Computer Science, Universidad Carlos III de Madrid, Avda. Universidad 30, 28911 Leganés, Madrid, Spain. (e-mail: {pperis, jestevez}@inf.uc3m.es)

N. Sklavos is with Computer and Informatics Engineering Department, Technological Educational Institute of Western Greece, Greece (E-mail:nsklavos@ieee.org)

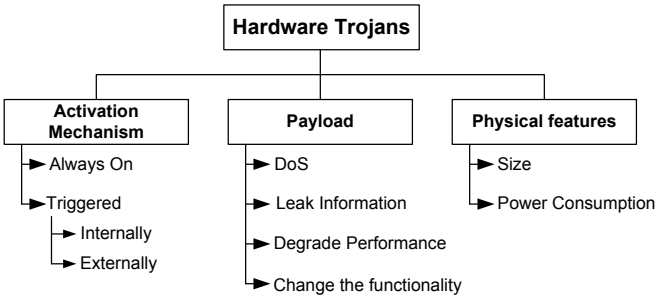

Fig. 1. Trojan taxonomy based on trigger, payload and physical features

The rest of the paper is organized as follows. In Section II properties of Hardware Trojans are studied. In Section III typical TRNG constraints are considered in relation to Hardware Trojans. In section IV we present an example of a simple but effective Hardware Trojan. Finally, conclusions are drawn in Section V.

II. Hardware Trojans

As aforementioned in the introduction, malicious modification of hardware (Hardware Trojans) during design or fabrication has emerged as a major security concern. These modifications cause an integrated circuit to have altered functional behaviour, potentially with disastrous consequences in safety-critical applications [9].

There are several methods of classifying hardware Trojans reported in the literature [10][11][12]. These methods use various characteristics in order to categorize the different Hardware Trojans. For example, in [10], the authors classify different Hardware Trojans regarding their activation mechanism (combinational and sequential). In [11], Hardware Trojans are classified based on three attributes: physical, activation and action. A very rich taxonomy where many features are considered can be found in [12].

The classification presented in [11] is sufficient for a first approximation to the integration of Hardware Trojans in TRNGs. We have studied the characteristics sketched in Fig. 1 and related them with the common TRNG demands. Our final goal will be to have some effects in the TRNG output without being detected by the on-line tests.

A. Activation Mechanism

Trojans can be classified based on two subsets regarding their activation mechanism. Trojans that are always active as soon as the IC is powered on are classified as Always-On Trojans. On the other hand, triggered Trojans subset comprehends the Trojans that need a special condition to be triggered. This kind of Trojans can be further classified regarding the

kind of trigger: Internal or External. Internal triggered Trojans typically wait a certain time or a specific sequence of rare logic values at internal nodes to be activated. Physical-condition-based Trojans that activate the malicious hardware by certain events (e.g. Temperature) are very interesting from the point of view of controllability. On the other hand, external triggered Trojans needs a specific combination of inputs or outputs to be activated.

Regarding the suitability of Hardware Trojans for TRNGs, Always-On Trojans could be discarded because they could be detected more easily by the on-line tests that the TRNGs typically embed. It would be desirable to have an activation mechanism that allows an attacker to control when the malicious hardware is operating. This feature grants an adversary the opportunity to have an influence over a specific key generation process or bypass some on-line tests that are executed from time to time.

B. Payload

Hardware Trojans are also classified regarding their intended behaviour. We have considered four possible categories as in [12]: Denial of Service (DoS), leak information, performance degradation and functionality changes. As the name implies, DoS Trojans can cause the denial of service for a requested service at a specific time. Other possible payload is the leakage of the secret key or other useful information for an attacker. Some Trojans can degrade the performance of their target. In the particular case of TRNGs a degradation of the performance can be translated into a lower statistical quality at the output. Finally some malicious hardware provoke changes in the intended functionality.

The effects of Hardware Trojans in TRNGs have to be considered with caution. An effective Hardware Trojan should not be detected by the on-line tests. For this reason Hardware Trojans whose effects are DoS or changes into its functionality would be easily detected . On the other hand a Hardware Trojan that can leak the random number generated by the TRNG could be very powerful since these random numbers are typically used as keys in cryptographic systems. This kind of Trojans typically affects the communications channels (Scan Chain, RS232, etc.) and for that reason these ones are not under the scope of this paper. Finally a desirable payload could be a slightly degradation of the statistical quality at the output of a TRNG but being undetectable for the on-line tests.

C. Physical Characteristics

Trojan physical characteristics represent various hardware manifestations. These hardware manifestation can compromise the undetectability of a Hardware Trojan. For this first approximation to the topic we have only considered two physical features: Size and Power consumption. The Hardware Trojan size is very important in order to avoid detection by simple visual inspection. Typically a malicious piece of hardware passes undetected due to the high placement density of circuits. Regarding power consumption, it is crucial that the power consumed by the Hardware Trojan is negligible in comparison with the rest of the circuit.

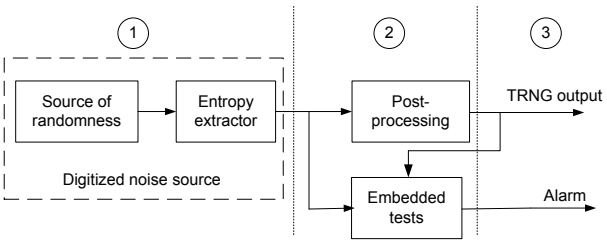

Fig. 2. General scheme of a TRNG

In the specific case of TRNGs, as we will see later, there are many TRNG proposals using very different resources. For TRNGs with a low logic count [13] (9 inverters + 6 registers) is almost impossible to pass undetected by simple visual inspection. On the other Hand, there are many other TRNGs that demand a bigger footprint (e.g.,[14]) and could contain a malicious hardware. Regarding the power consumption, TRNGs generally consume a lot of power. An example of these TRNGs are those that exploit the jitter to extract randomness; jittery clocks are usually used as entropy sources. These jittery clocks are mainly composed of Ring Oscillators (ROs), Self-Timed Rings (STR) or Phase-Locked-Loop (PLL) which are very demanding in terms of power. For this, the consumption of the Hardware trojan would be negligible and would render undetectable this piece of hardware.

III. True-Random Number Generators

True Random Number Generators are one of the basic cryptographic blocks used in security systems. These generators are commonly used to generate secret keys, and for that reason, they are one of the most important elements in a cryptographic system.

TRNGs commonly use some kind of physical phenomenon as a source of entropy. Typically, these phenomena are analogue, therefore, some extraction mechanism is needed in order to convert the analogue values in digital ones. Once the entropy source has been digitized, the statistical properties of the digitized signal will be evaluated with the purpose of establishing the TRNG quality. After this first evaluation, it is often conclude that a post-processing block is required to correct the output distribution. Finally, due to the importance of the TRNGs in security systems, it is recommendable to check the quality of the random output during its generation. Some embedded tests are employed to set an alarm when the generated output do not comply with some statistical requirements. The typical blocks used in embedded TRNGs are depicted in Figure 2.

A. Typical On-line Tests

In terms of security the idea of implementing embedded tests that evaluate the quality of the TRNG (mandatory in AIS31 [7]) is very interesting. As explained in Section II, our final goal is to have some effects in the TRNG output without being detected by these mentioned on-line tests.

Among the most well-known battery tests stand out ENT [4], DIEHARD [5], NIST [6] and AIS31 [7]. These battery

tests are used to carry out statistical evaluations on a TRNG output (i.e., stream of 1s and 0s). These tests allow a little bias in the output. The objective of the Hardware Trojan will be to try to exploit this fact by increasing the bias while passing all the tests.

It is important to remark that these tests do not guarantee true-randomness. For example if the TRNG is replaced by a high quality PRNG, it would be unfeasible for standard statistical tests to distinguish between a true-random number and a pseudo-random number. We start out on the assumption that we are trying to integrate a malicious hardware in a TRNG.

B. Hardware Trojan influence zone

We have divided a general TRNG scheme in three zones (see Fig. 2) where a Hardware Trojan can have an influence. In this subsection we analyse briefly the advantages and disadvantages of integrating the malicious Hardware in each zone.

A Hardware Trojan that affects the entropy source (zone 1) probably will be detected by on-line tests that evaluate the statistical properties of the TRNG raw output. In addition, some authors have presented on-line tests to check some parameters of the entropy source, in order to guarantee that their designs generate true-random values e.g., in [15], V. Fischer et al. proposed a simple precise method for measuring the jitter that is closely linked to the entropy of the generated bit stream, which depends on the size of the jitter. In conclusion, integrating a Hardware Trojan that affects the entropy source is a challenging task because up to three stages of on-line tests have to be passed.

Following the same idea, a malicious hardware that affects zone 2 will need to pass up to two statistical tests (before and after the post-processing stage). Moreover, the effects of the Hardware Trojan will be mitigated by the post-processing, even annulling its effect depending on the post-processing technique used.

All in all it seems to be the best option to integrate the Hardware Trojan in zone 3 where only one on-line test has to be passed.

IV. Example

In this section we present a Hardware Trojan simulation that fulfils with the above mentioned features. In particular, the example consists of a XOR gate, an AND gate and 2 ROs shifted in phase 180^o (see Fig. 3). The proposed Hardware Trojan is based on the Markettos et al. attack against ROs [16]. In this work, the authors are able to control the phase of several ROs placed in the same device by injecting a sine wave in the power supply. We have exploited the same phenomenon for triggering the malicious hardware.

The operation of the circuit is as follow. The output of the ROs are XORed generating a logical '1' while the Hardware Trojan is not triggered –TRNG ouput is not altered. Once the Hardware Trojan is triggered, the two ROs will be in phase. The XOR result will be a logical '0' that means that the final output will be '0' –that is, the output provided by the TRNG is discarded. It is very important to affect only a few bits in order to pass the statistical tests.

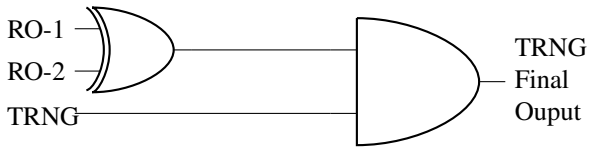

Fig. 3. General scheme of the proposed Hardware Trojan

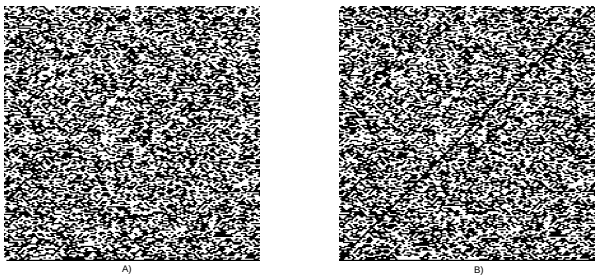

Fig. 4. Hardware Trojan Simulation

We have tested this example using Simulink (MathWorks). The ROs has been simulated using Pulse generators blocks. For the TRNG we have used a random source block, which passes randomness tests. The unaltered bit-stream generated by the random block is depicted in Fig. 4(a). On the other hand, in Fig. 4(b), the Hardware Trojan is activated several times generating a slight bias. The effect can be appreciate in the diagonal of the picture but it is undetectable through the on-line statistical tests –the manipulated data stream keeps passing the tests.

In order to show the lightweightness of this example, we have implemented the Cherkaoui et al. proposal for 511 stages ([14]) on an Spartan-3E. In Fig. 5(a), the TRNG free of Trojans is depicted. In Fig. 5(b) the aforementioned Trojan has been integrated in the design. Finally, in Fig. 5(c), the malicious Hardware has been highlighted.

In summary, these are the features of the proposed TRNG Hardware Trojan:

- Trigger: Frequency Injection.
- Payload: Generating a slight bias in the final output.
- Physical Characteristics: Very lightweight.
- TRNG Influence Zone: Zone 3.

V. Conclusions

TRNGs are often integrated into security chips for the generation of random or pseudo random cryptographic tokens. If this value is compromised, the security of the whole security system would be jeopardized independently whether the other system elements (e.g., cipher or hash function) are secure. To evidence this serious threat, to the best of our knowledge, we have presented the first Hardware Trojan inserted into a TRNG. The malicious piece of hardware is undetectable for the main point of views. On the one hand, the Hardware Trojan is easily camouflaged with the rest of components and can not be detected by visual inspection. On the other hand, there is not indication of its presence analysing the results of the

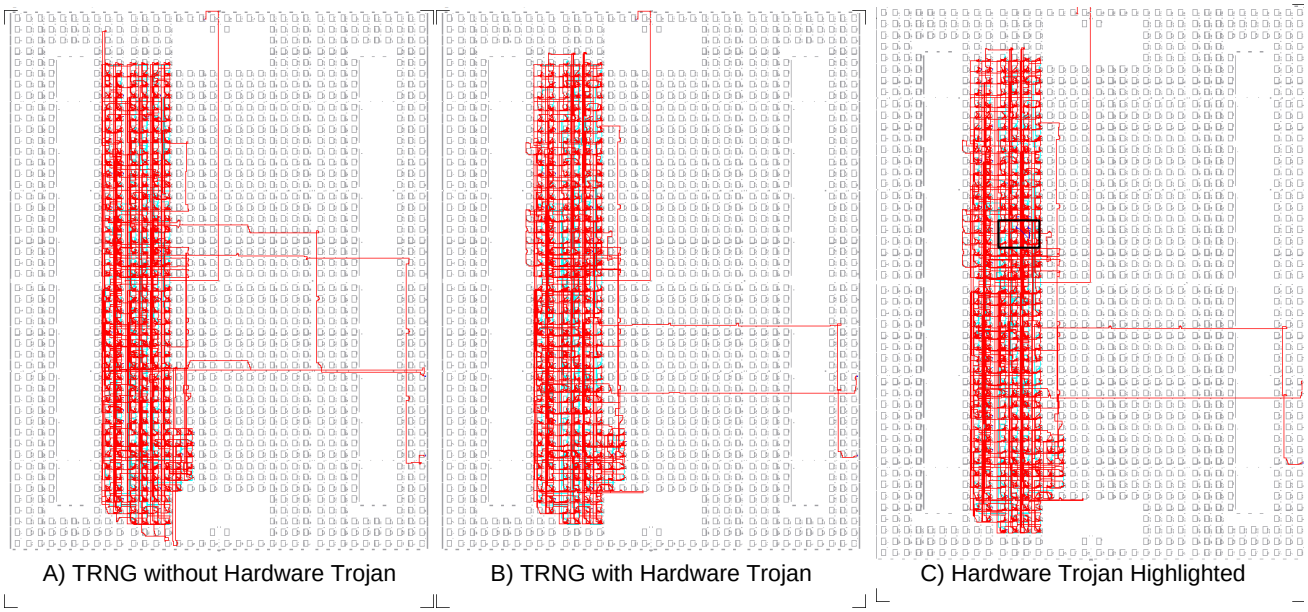

Fig. 5. TRNG and Hardware Trojan implemented in an FPGA

embedded on-line test (randomness evaluation). Apart of this, the adversary have control to fix some values of the output to zero during short intervals, which could be used as the start of an attack in which these values (all zeros) are used –e.g., the adversary might know that certain bits of the secret key are zero.

Summarizing it is real that Hardware Trojans can be incorporated in a TRNG. We have shown an approach to success in this attempt but many other approaches could be used. We hope this paper server as seed to many other investigations in the area.

REFERENCES

[1] "Breakthrough silicon scanning discovers backdoor in military chip," in *Cryptographic Hardware and Embedded Systems CHES 2012*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds., 2012, vol. 7428.

[2] G. Becker, F. Regazzoni, C. Paar, and W. Burleson, "Stealthy dopant-level hardware trojans," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, ser. Lecture Notes in Computer Science, G. Bertoni and J.-S. Coron, Eds. Springer Berlin Heidelberg, 2013, vol. 8086, pp. 197–214. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40349-1-12

[3] T. Sugawara, D. Suzuki, R. Fujii, S. Tawa, R. Hori, M. Shiozaki, and T. Fujino, "Reversing stealthy dopant-level circuits," in *Cryptographic Hardware and Embedded Systems CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds. Springer Berlin Heidelberg, 2014, vol. 8731, pp. 112–126. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44709-3-7

[4] J. Walker, "Randomness battery," 1998. [Online]. Available: http://www.fourmilab.ch/random/

[5] G. Marsaglia, "The marsaglia random number cdrom including the diehard battery of tests of randomness," 1996. [Online]. Available: http://stat.fsu.edu/pub/diehard

[6] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *Technical Report*, 2001.

[7] W. Schindler and W. Killmann, "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '02. London, UK, UK: Springer-Verlag, 2003, pp. 431–449. [Online]. Available: http://dl.acm.org/citation.cfm?id=648255.752732

[8] J. Francq, "Hardware trojans detection methods," in *Training School on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE)*, July 2014.

[9] R. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International*, Nov 2009, pp. 166–171.

[10] M. Banga and M. Hsiao, "A region based approach for the identification of hardware trojans," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, June 2008, pp. 40–47.

[11] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, June 2008, pp. 15–19.

[12] H. Salmani and M. Tehranipoor, "http://trust-hub.org/resources/benchmarks." in *Trust HUB.org*.

[13] E. Bohl and M. Ihle, "A fault attack robust trng," in *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, June 2012, pp. 114–117.

[14] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A Very High Speed True Random Number Generator with Entropy Assessment," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, ser. Lecture Notes in Computer Science, G. Bertoni and J.-S. Coron, Eds. Springer Berlin Heidelberg, 2013, vol. 8086, pp. 179–196.

[15] V. Fischer and D. Lubicz, "Embedded evaluation of randomness in oscillator based elementary trng," in *Cryptographic Hardware and Embedded Systems CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds. Springer Berlin Heidelberg, 2014, vol. 8731, pp. 527–543. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44709-3-29

[16] A. T. Markettos and S. W. Moore, "The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators," in *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 317–331.