TOTAL: TRNG On-the-fly Testing for Attack detection using Lightweight hardware

Bohan Yang*, Vladimir Rožić*, Nele Mentens*, Wim Dehaene† and Ingrid Verbauwhede*
* ESAT/COSIC and iMinds, KU Leuven,
† ESAT/MICAS, KU Leuven and IMEC,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email:{bohan.yang, vladimir.rozic, nele.mentens, wim.dehaene, ingrid.verbauwhede}@esat.kuleuven.be

***Abstract*—We present a design methodology for embedded tests of entropy sources. These tests are necessary to detect attacks and failures of true random number generators. The central idea of this work is to use an empirical design methodology consisting of two phases: collecting the data under attack and finding a useful statistical feature. In this work we focus on statistical features that are implementable in lightweight hardware. This is the first paper to address the design of on-the-fly tests based on the attack effects. The presented design methodology is illustrated with 2 examples: an elementary ring-oscillator based TRNG and a carry-chain based TRNG. The effectiveness of the tests was confirmed on FPGA prototypes.**

I. Introduction

True random number generators (TRNG) are essential components in security systems. They are used for generating keys, challenges and masks, and as such they are often the most vulnerable part of the secure system. There are numerous cases of security threats in systems using strong cryptography where security was compromised due to a failure of the random number generator [1]. In order to provide the security of the generated numbers, the TRNG has to be subjected to extensive testing, not only after manufacturing but also during the operation.

Several implementations of on-the-fly tests for randomness are available for both FPGA and ASIC platforms. Related work can be divided into two groups. The first group consists of implementations of standard statistical tests [2], [3], [4], [5], [6], [7], [8], [9]. These tests are originally designed for prototype evaluation, so some of them are too computationally intensive to be practical for on-the-fly testing. All of these are generic black-box tests, and their usefulness depends on the used entropy source and the type of the attack. The second group consists of dedicated statistical tests based on the stochastic model of the entropy source, such as [10] and [11]. The problem with this approach is that not every TRNG is supported by a stochastic model, and existing stochastic models rely on many simplifications and assumptions about the mechanism of the physical randomness-generating process.

This work is supported in part by the funding of the Research Council KU Leuven (C16/15/058), the Flemish Government through FWO G.0550.12N, G.0130.13N and FWO G.0876.14N, the Hercules Foundation AKUL/11/19, and by the European Commission through the Horizon 2020 research and innovation programme under grant agreement No 644052 HECTOR. In addition, this work was supported in part by the Scholarship from China Scholarship Council (No.201206210295).

Considering that there are a few well-known attack vectors (i.e. voltage, frequency, temperature...), in our opinion, it is more efficient to design on-the-fly tests based on the effects of these attacks. The compromised output of an entropy source under attack is more likely to manifest itself as a change in a simple statistical feature rather than a complex and intricate one. To avoid overly extensive testing (which comes with a penalty in increased design area and energy consumption) it is important to know which type of statistical defects appears in a realistic attack scenario.

This paper presents a design methodology called TOTAL (TRNG On-the-fly Tests for Attack detection using Lightweight hardware). We suggest using an empirical design procedure consisting of the data collection and a search for the useful statistical features. We recommend to test the entropy source under the normal operating conditions and under different attacks such as under-powering, oversampling, glitching and temperature attacks. The sequences of the produced bits should be recorded. The next phase consists of applying different statistical features on the collected data and finding a useful feature for distinguishing between the normal operation and the attack. In this context, the usefulness of the feature is evaluated according to the attack detection probability. We focus on the features that have compact hardware implementations. We created an extensive list of these features including: the number of ones in a sequence, the auto-correlation coefficients, frequencies of different patterns and the number of runs of a predefined length. Once useful features are found, the tests are implemented by setting the appropriate threshold values. This methodology enables the designer to specify the length of a sequence under test and the false alarm trigger level, as well as to estimate the usefulness of the test. The effectiveness of the proposed methodology is higher when more features and more attacks are tested. The attacks that don't change any of the features are not likely to create any harm. This is the first paper to address the design of on-the-fly tests based on the attack effects.

This paper is organized as follows. In Section 2 we present a step-by-step design methodology for on-the-fly testing modules in lightweight hardware. This methodology is illustrated with two examples provided in Sections 3 and 4. In Section 3, we present a case study of an elementary ring-oscillator based

978-3-9815370-6-2/DATE16/

entropy source implemented on a Xilinx Spartan-6 FPGA and subjected to oversampling and under-powering attacks. Useful features are detected and hardware tests are designed and validated. In Section 4 we present a case study of the high-throughput TRNG based on carry-chains [12]. Finally, in Section 5 we present our conclusions and proposals for future work.

II. Design Methodology

A. Background

A true random number generator consists of the entropy source, the post-processing module and the on-the-fly tests. The entropy source is the only component that generates new randomness. It relies on an unpredictable physical process such as timing jitter or thermal noise to produce unpredictable digital output. The bits produced by the entropy source are called *raw bits*. These bits don't have to be perfectly random because the post-processing module is used to compress these data into a full-entropy (perfectly random) output. On-the-fly tests are used to monitor the quality of the produced raw bits and to activate an alarm when attacks and weaknesses are detected. These tests are usually implemented as statistical tests. They look for the patterns and regularities in the generated bits and set an alarm signal when statistical defects are detected. These tests have both false positive (Type I) and false negative (Type II) errors.

B. Design steps

The proposed design methodology for on-the-fly tests relies on extensive testing of the entropy source under various operating conditions to simulate all practical attack scenarios. The goal is to find one or more statistical features that are suitable for attack detection and that are also implementable using lightweight hardware. Another design goal is low length of the test sequence. This requirement is needed because of the requirement to stop the TRNG output when an attack or a weakness is detected. If a statistical test is used on a sequence of length n, then by the time a weakness is detected, n compromised bits are sent to the output. The only way to prevent compromising the output is to buffer the generated data and send them to the output only if the data passed the test. This solution is practical only for short sequences which don't require much storage.

Since these tests are tailored for the specific entropy source, it is possible to achieve a better performance compared to the universal statistical tests. This methodology aims for compact hardware implementations, short bit sequences, low computational latency and reliable attack detection.

The proposed design methodology consists of the following steps:

1) Data Collection
2) Preliminary detection of useful features
3) Feature verification
4) Attack effort analysis
5) HW implementation
6) HW Validation

Here we explain each step in more detail:

a) Data Collection: Firstly, data produced by the entropy source (the raw bits) should be collected under the normal operating conditions. We call these data the golden reference. We suggest collecting data sets of 512 consecutive bits. In our experience, this length is enough to design useful, robust tests that are significantly faster than generic statistical tests. For the preliminary analysis, we recommend collecting around 8000 sequences.

Afterwards, the entropy source should be exposed to different attacks and new sets of data should be collected. Again, we recommend collecting around 8000 sequences of 512 bits for each attack. At this step it is important to include all low-cost attacks such as changes in temperature, voltage and clock frequency. The security of the method depends on the extensiveness of the testing.

Each attack can be performed with different effort, for example the under-power attack can be performed using different voltage levels. In this phase, we are only collecting data using the highest attack effort such as the lowest voltage level for the under-power attack or the highest frequency for the oversampling attack. The collected data will be used to detect the useful statistical features that will be used in the test design.

b) Preliminary detection of useful features: Once the first set of data is collected, the goal is to compute the statistical features. We provide a recommended list of features that can be implemented in a compact manner:

1) Auto-correlation coefficients

$$C_i = \sum_{k=1}^{n-i} a_k \oplus a_{k+i} \tag{1}$$

where $a_1..a_n$ are the bits of the test sequence. Auto-correlation coefficients are computed for $i = 1...16$.

2) Number of ones in a sequence - N_1.
3) Number of overlapping 2, 3 and 4-bit templates in a sequence - $N_{00}...N_{11}, N_{000}...N_{111}, N_{0000}, N_{0001}...N_{1111}$.
4) Number of non-overlapping 2, 3 and 4-bit templates in a sequence $n_{00}...n_{11}, n_{000}...n_{111}, n_{0000}, n_{0001}...n_{1111}$.
5) Maximal Random Walk Divergence from Zero - S_{max}

$$S_i = \sum_{k=1}^{i} (2 \cdot a_k - 1) \tag{2}$$

$$S_{max} = max_{k=1..n}|S_k| \tag{3}$$

6) Number of runs of fixed length - $r_1...r_8$
7) Length of the longest run of the same bit value - r_{max}.

The features were selected based on the previous implementations of on-the-fly tests. We looked into the implementations presented in [2], [3], [4], [5], [6], [7], [8], [9] and selected the features that can be implemented in a simple and compact manner. In addition, we recommended using the auto-correlation coefficients. This list is not comprehensive and other features may be included. Each of these features can be implemented using one counter and a small amount of logic gates.

Each feature is computed for each of the collected sequences and the probabilities of the distributions are studied. Useful features are selected based on two criteria: the distribution of the golden reference, and the distinguishability of the distributions during attack. Only the features that show the normal distribution for the golden reference are considered. If no such features are found the data collection should be repeated using sequences longer than 512 bits. For each of the selected features the mean μ and the standard deviation σ are computed.

Further selection is done by evaluating the usefulness of features for attack detection with respect to type II error probability. We define usefulness as the probability of detecting an attack: $1 - P_2$ where P_2 is the probability of a type II error (false negative). We explore usefulness with respect to two types of tests: a sensitive test and a robust test. We define a sensitive test as a test with 1% type I error probability. This means that one in 100 sequences will trigger the alarm when the entropy source is not under attack. This test is implemented by checking if the measured feature is within the $(\mu - 2.58\sigma, \mu + 2.58\sigma)$ interval. The robust test is defined as a test with 10^{-9} type I error probability. It is implemented by checking if the measured feature is within the $(\mu - 6.11\sigma, \mu + 6.11\sigma)$ interval.

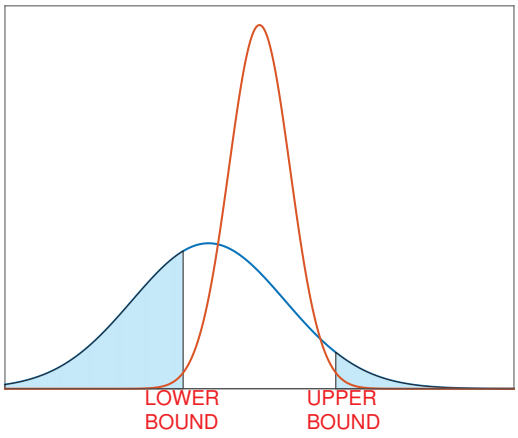
Other tests may be defined depending on the design specifications. The usefulness of each test is estimated as the ratio of sequences under attack that are detectable by the test.

$$usefulness = \frac{\#\text{Sequences outside of the range}}{\#\text{All sequences}} \quad (4)$$

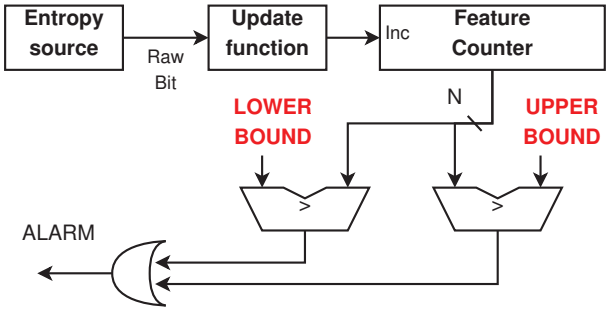
The features are ranked according to the computed usefulness for both types of test and for all attacks.

c) Feature verification: At this phase, the top ranked features for the sensitive test and the robust test are selected. More data is collected to verify the choice of the selected features. We suggest collecting 128000 sequences of 512 bits. These data are used to verify that the golden reference remains normally distributed and that detected features remain useful. If problems with the distribution of the golden reference are detected, all previous steps should be repeated using longer bit sequences or using different design parameters of the entropy source. If the usefulness is too low, then the features with lower rank are explored. If no useful feature is found then previous steps should be repeated using longer sequences or different parameters of the entropy source. In the end, two features are selected for each attack, one for the sensitive and one for the robust test. There may be some overlap between the features, for example when one feature is useful for detecting 2 attacks.

d) Attack Impact Analysis: In previous phases, we have used data under the highest attack impact. At this phase, we evaluate how different levels of attack impact affect the usefulness of the test. Additional data should be collected using different levels of the attack impact such as different voltage levels (for underpower attacks) or different frequencies (for oversampling attacks). Test usefulness should be computed for each level of the attack impact.



(a) Determining bounds for on-the-fly tests



(b) The generic architecture of the on-the-fly test

Fig. 1: The principle for hardware implementation

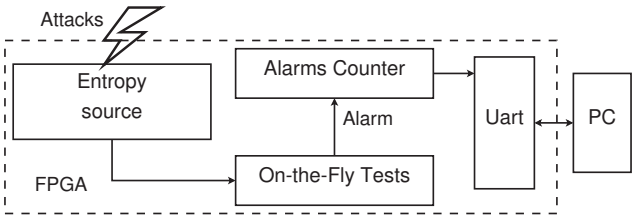


Fig. 2: Platform for hardware validation

e) HW Implementation: As a general rule, tests with a high false positive (Type I) error rate are more efficient for attack detection. Type I error rate is a design parameter which should be set based on the application requirements. If the maximal allowed error rate is once in a lifetime of the product (for example when a manual reset is required whenever an alarm signal is detected), as required by [13], then a robust test with very low error rate should be implemented. The usefulness of this test for attack detection may be very low. For some applications, sensitive tests may be more appropriate. If the application doesn't require manual reset after attack detection then a test with higher Type I error rate (for example 1%) may be appropriate. Such test can detect attacks with higher success rate.

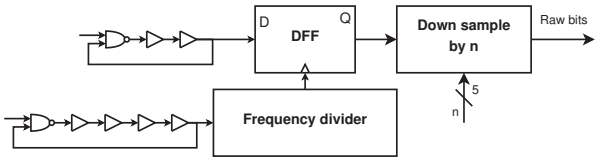


Fig. 3: Elementary TRNG

The generic architecture of the on-the-fly test is shown in Figure 1b. Raw bits from the entropy source are sent to the update function which generates the Inc signal, for example in the template matching tests, the signal will be activated whenever a specific pattern is detected. This signal is used to increase the counter. At the end of the sequence, the counter value is compared to the predefined lower bound and the upper bound. These bounds are computed based on the required error rates and the parameters of the golden reference Gaussian distribution. If the counter value is outside of the range, the alarm signal is activated. The example given in Figure 1a shows the counter distributions under normal conditions (orange line) and under attack (blue line). The bounds are shown for 1% type I error rate. The shaded area under the curve is equal to the usefulness of the test.

f) HW Validation: As a last step of the methodology, the hardware implementation should be evaluated. The platform shown in Figure 2 is used for this purpose. The entropy source under normal operating conditions is used to generate 1024000 sequences. The embedded test evaluates a sequence and produces an alarm signal if the test fails. A counter keeps track of the alarm signals. After all sequences are tested, the counter value is sent to the output and the error rate is computed. The purpose of this validation is to verify if the error rate is within the predicted limits. The same procedure is repeated for the entropy source under attack and the attack usefulness is computed. Validation should be done for all attacks and all implemented tests.

III. Case study I: Elementary TRNG

In this section we provide an example to illustrate the proposed design methodology. We use a simple elementary true random number generator based on ring oscillators and we apply our methodology to design a test for detection of the oversampling attack. In a commercial product, more tests to detect attacks should be included (such as interlock attacks, glitching attacks and the temperature attacks).

A. TRNG architecture

The elementary TRNG consists of a fast oscillator implemented using 3 look-up tables (LUTs) sampled by a slow oscillator implemented using a 5-stage LUT-based ring oscillator and a frequency divider. This TRNG relies on the timing jitter for generating entropy. For longer jitter accumulation time, the output approaches the full-entropy. The main problem of this generator is that the timing jitter accumulates very slowly. In order to achieve more than 0.99 bits of entropy per output bit, the output needs to be down-sampled by a factor of 32.

TABLE I: Features with 100% usefulness for the elementary TRNG

Sensitive tests	Robust tests
C_1, N_{00}, N_{01}, N_{10}, N_{000},N_{010},N_{101},N_{111}, N_{0000}, N_{0010}, N_{0100}, N_{0101}, N_{1010}, N_{1011}, N_{1101}, n_{01}, n_{10}, n_{010}, n_{101}, n_{0010}, n_{0100}, n_{1011}, n_{1101}	C_1, N_{01}, N_{10}, N_{010}, N_{101}, n_{01}, n_{10}, n_{010}, n_{101}, n_{1011}, n_{1101}

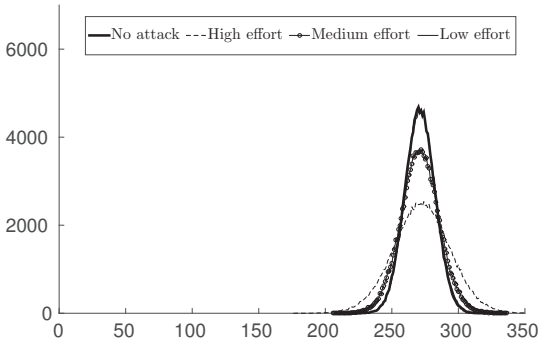
B. Test design

The over-sampling attack is simulated by reducing the down-sampling rate. As shown in Figure 3 we have provided an additional input for selecting the down-sampling factor. The design was evaluated using three down-sampling factors (2, 4 and 8) corresponding to a high, medium and the low attack effort respectively. For the initial analysis only the highest attack effort was used.

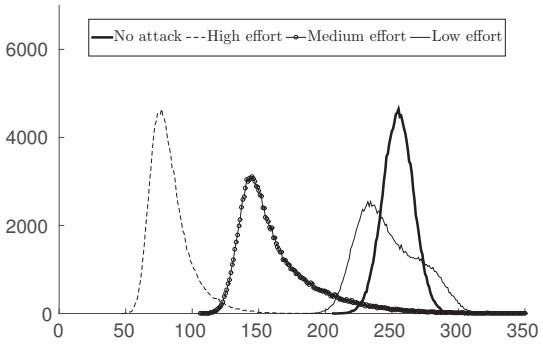
The usefulness of the features was evaluated with respect to the sensitive test (corresponding to 1% type I error rate) and the robust test (10^{-9} error rate). The features are ranked based on the measured usefulness for both tests and features with maximal usefulness are shown in Table I. The overlapping 4-bit template N_{0101} and the auto-correlation coefficient C_1 were chosen for implementations. Any feature listed in the table could be used for test design.

Additional data was collected to justify the choice of the features and to analyze the test performance under different attack impacts. Figure 4 shows examples of a useful and a useless feature. Additional 128000 sequences were collected under normal operating conditions and under high, medium and low attack effort. The features N_1 and C_1 were computed for each sequence and the corresponding distributions are shown. Figure 4a shows the distribution of the number of ones within the sequence. As can be seen this feature is not useful because the distributions under attack are centered around the same mean value as under normal operating conditions which means that the attack doesn't significantly affect the bias. Standard deviations seem to increase with higher attack effort but not enough to be useful for test design. On the other hand, the auto-correlation coefficient C_1 is very useful for attack detection. The mean of the distribution shifts to the left with higher attack effort as shown in Figure 4b. Under highest attack effort the distribution is completely distinguishable from the original one resulting in almost perfect usefulness of the test. This usefulness is slightly lower for lower attack effort as expected.

We tried to perform the underpower attack using the same methodology. Our implementation platform uses the core power supply voltage of $1.2V$. For voltages below $0.86V$ the communication interface stopped working. We have collected data for different voltage values between 0.86 V and 1.2 V but couldn't find any useful features.



(a) N_1



(b) C_1

Fig. 4: Features distributions

TABLE II: Hardware utilization results for the elementary TRNG

	FPGA			ASIC
	Slices	LUTs	FFs	(GE)
Sensitive test	9	28	25	245.75
Robust test	10	26	22	233.5
Combined test	14	42	35	361.5

C. Implementation and validation

We have implemented both the sensitive (1% error rate) and the robust test (10^{-9} error rate). The sensitive test was implemented using an overlapping 0101 template. The test verifies if the number of templates in the sequence is within the $[17, 47]$ interval. The robust test is implemented using one XOR gate and a counter for computing C_1. Comparators are used to check if C_1 is within the $[183, 326]$ interval. Table II shows the hardware utilization results. All proposed hardware designs are synthesized using Xilinx ISE14.7 on Spartan-6 XC6SLX45 FPGA and Synopsys Design Compiler D-2010.03-SP4 to UMC's $0.13\mu m$.1P8M Low Leakage Standard cell Library.

The usefulness of the implementation was experimentally verified using 1024000 sequences. The results are reported in Table III. For high attack impact both tests have very high usefulness. For medium attack impact, the usefulness is still very high but it is significantly lower for the low attack impact. The measured Type I error is within the expected boundaries (0.93% for the sensitive test and 0% for the robust test).

TABLE III: Test validation for the elementary TRNG

	sensitive test usefulness (%)	robust test usefulness (%)
Low attack impact	34.85	≈ 0
Medium attack impact	99.67	83.97
High attack impact	100	99.84

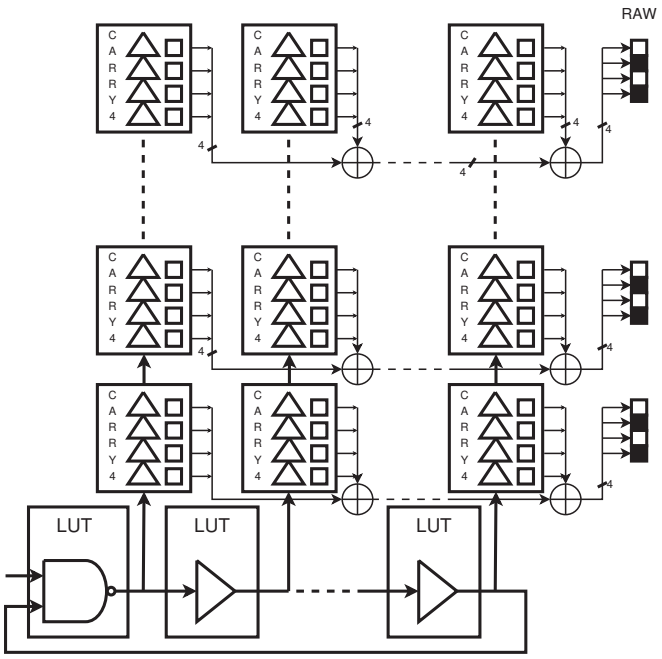


Fig. 5: Carry-chain based TRNG

The robustness of the methodology with respect to process-variations was evaluated. The validation platform was implemented on a second FPGA of the same type using the same bit-stream. The experiments were repeated and the results showed that type I error rate remains constant over different FPGAs of the same type. Upon further investigation, it was found that the higher oversampling rate was needed to achieve the same attack impact.

IV. Case study II: Carry-chain TRNG

In this section we apply the TOTAL methodology to design on-the-fly tests for a carry-chain based TRNG on FPGA [12].

A. TRNG architecture

Figure 5 shows the architecture of the implemented entropy source. Random jitter accumulated in the ring oscillator is extracted using tapped delay lines built with Xilinx carry-chain primitives. The clock frequency is $100MHz$ and a sample is taken every 32 clock cycles. Shorter accumulation time results in lower entropy per bit.

TABLE IV: Top features for the carry-chain based TRNG

sensitive tests		robust tests	
Features	usefulness (%)	Features	usefulness (%)
N_{111}	90	C_1	35
C_2	83	N_{01}	35
N_{1110}	82	N_{10}	35
n_{111}	82	N_{010}	33
N_{0111}	82	r_1	32

TABLE V: Hardware utilization results for the carry-chain based TRNG

	FPGA			ASIC
	Slices	LUTs	FFs	(GE)
Sensitive test	10	26	24	239.5
Robust test	10	26	22	233.5
Combined test	14	40	34	353.25

B. Test design

The oversampling attack was performed by sampling the entropy source in every clock cycle. The collected data didn't show any detectable weaknesses. The under-power attack, when applied on the original design also didn't result in any significant changes. However, the combination of the two attacks with the voltage reduced to $0.9V$ and the sample taken in each clock cycle resulted in some useful features. The ranking of the features with respect to the sensitive test and the robust test as defined in the previous section is shown in Table IV. The overlapping 3-bit template N_{111} and the auto-correlation coefficient C_1 are the most useful features for the sensitive and the robust test respectively.

C. Implementation and validation

Both test implementations are very compact as shown in Table V. Each individual implementation consumes 10 slices on a Xilinx Spartan-6 FPGA and the combined implementation of both tests consumes only 14 slices. The ASIC implementation is also very compact with less than 240 GE for individual implementations and around 350 GE for a combined implementation.

The FPGA implementation was used to evaluate the type I error rate and the usefulness. The measured type I error rate is within the expected boundaries (0.65% for the sensitive test and 0% for the robust test). The measured usefulness is 44.36% for the robust test and 89.4% for the sensitive test.

V. Conclusions and Future Work

In this paper, we have proposed a design methodology for dedicated, lightweight, low-latency on-the-fly tests for monitoring the quality of entropy sources. This methodology relies on experimental results to find the optimal statistical features for designing on-the-fly tests. This experiment-oriented approach guarantees that the tests are useful for detecting attacks. The security of the proposed method depends on the richness of the pool of statistical features and the number of attacks applied during the design phase. The main benefit of the proposed method over using standard statistical tests is the improved performance in terms of latency, area and test effectiveness. This benefit comes from the fact that the produced tests are tailored for a specific entropy source.

The proposed methodology was applied to design dedicated tests for two entropy sources. Lightweight tests were designed and their effectiveness for attack detection was confirmed by experiments.

The presented work can be extended in several directions in the future. The list of recommended statistical features can be extended with other features that can be implemented in a compact manner. In addition, the methodology should be extended to account for different combinations of features rather than the current single feature based testing. The experimental work can be extended to include more entropy sources and more attacks. The final goal is to provide a list of attacks that are effective for different entropy sources implemented on various platforms and a list of lightweight tests to detect these attacks.

References

[1] D. J. Bernstein, Y. Chang, C. Cheng, L. Chou, N. Heninger, T. Lange, and N. van Someren, "Factoring RSA keys from certified smart cards: Coppersmith in the wild," in *Advances in Cryptology - ASIACRYPT*, 2013, pp. 341–360.

[2] R. Santoro, O. Sentieys, and S. Roy, "On-line monitoring of random number generators for embedded security." in *ISCAS*. IEEE, 2009, pp. 3050–3053.

[3] D. Hoţoleanu, O. Creţ, A. Suciu, T. Gyorfi, and L. Văcariu, "Real-time testing of true random number generators through dynamic reconfiguration," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, sept. 2010, pp. 247 –250.

[4] A. Vaskova, C. Lopez-Ongil, A. Jimenez-Horas, E. S. Millan, and L. Entrena, "Robust cryptographic ciphers with on-line statistical properties validation," in *11th IEEE International On-Line Testing Symposium (IOLTS)*, 2010, pp. 208–210.

[5] A. Vaskova, C. López-Ongil, E. San Millán, A. Jiménez-Horas, and L. Entrena, "Accelerating secure circuit design with hardware implementation of DIEHARD battery of tests of randomness," in *12th IEEE International On-Line Testing Symposium (IOLTS)*, 2011.

[6] F. Veljković, V. Rožić, and I. Verbauwhede, "Low-cost Implementations of On-the-fly Tests for Random Number Generators," in *2012 Design, Automation & Test in Europe*, 2012, pp. 959–964.

[7] V. B. Suresh, D. Antonioli, and W. P. Burleson, "On-chip lightweight implementation of reduced NIST randomness test suite," in *Hardware Oriented Security and Trust (HOST)*, 2013, pp. 93–98.

[8] B. Yang, V. Rožić, N. Mentens, W. Dehaene, and I. Verbauwhede, "Embedded HW/SW platform for on-the-fly testing of true random number generators," in *Proceedings of the 2015 Design, Automation & Test in Europe DATE*, 2015, pp. 345–350.

[9] B. Yang, V. Rožić, N. Mentens, and I. Verbauwhede, "On-the-fly Tests for Non-ideal True Random Number Generators," in *IEEE International Symposium on Circuits and Systems, ISCAS*, 2015, pp. 2017–2020.

[10] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *Journal of Cryptology*, vol. 24, no. 2, pp. 398–425, 2011.

[11] V. Fischer and D. Lubicz, "Embedded Evaluation of Randomness in Oscillator Based Elementary TRNG," in *CHES*, 2014, pp. 527–543.

[12] V. Rožić, B. Yang, W. Dehaene, and I. Verbauwhede, "Highly efficient entropy extraction for true random number generators on FPGAs," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, 2015, pp. 116:1–116:6.

[13] E. Barker and J. Kelsey, "Recommendation for the entropy sources used for random bitgeneration," ser. NIST DRAFT Special Publication 800-90B, 2012.