Lightweight Prediction-Based Tests for On-Line Min-Entropy Estimation

Miloš Grujić, Vladimir Rožić, Bohan Yang and Ingrid Verbauwhede

***Abstract*—Health tests (on-the-fly tests) play an important role in true random number generators because they are used to assess the quality of the bits produced by entropy source and to raise an alert when failures or attacks are detected. Most of classical tests are implemented as statistical tests. A set of new health tests based on predictors was presented by NIST (National Institute of Standards and Technology) in CHES 2015. These off-line tests attempt to predict the next output of the entropy source by trying to learn the patterns that the previously produced sequence of bits may possess. We provide the first integrated lightweight implementation of prediction-based tests for min-entropy estimation and verify their validity.**

***Index Terms*—Field-programmable gate arrays (FPGA), min-entropy estimation, True random number generators (TRNG).**

I. Introduction

TRUE random number generators (TRNG) are necessary components of every cryptographic system. They are most commonly used for generation of secret session keys, challenges, salts, nonces and padding values. The entropy source is the core component of TRNGs and relies on unpredictable physical events in order to produce unpredictable raw bits [2]. However, entropy sources are also very sensitive to external perturbations, making them an ideal target for the attacker of the crypto system. Therefore, the quality of the produced raw bits needs to be monitored in order to detect attacks and weaknesses of the entropy source.

A suite of new prediction-based tests for min-entropy estimation during prototype evaluation was proposed in [1]. Predictors are machine learning units that constantly update their internal state based on a success/failure of their previous prediction. In this light, min-entropy is defined as the most conservative way of predicting the most-likely output of the entropy source.

Our contribution is a flexible lightweight FPGA implementation of three prediction-based tests for estimation of min-entropy level. These implementations are shown to be suitable for on-the-fly tests and not only during prototype evaluation. Our implementations use short bit sequences produced by the entropy source, enabling attack or failure detection with very low latency.

The authors are with KU Leuven, imec-COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium (e-mail:{milos.grujic, vladimir.rozic, bohan.yang, ingrid.verbauwhede}@esat.kuleuven.be).

This work is supported in part by the funding of the Research Council KU Leuven: C16/15/058, the Hercules Foundation AKUL/11/19, and through the Horizon 2020 research and innovation programme under grant agreement No 644052 HECTOR and ERC Advanced Grant 695305.

II. Set of Implemented Predictors

The predictors proposed in [1] are divided into two categories based on the interpretation of data samples - *Categorical data predictors* and *Numerical predictors*. In *Categorical data predictors* samples serve as bare labels, while in *Numerical predictors* samples have numerical meaning. In order to simplify the implementations, data samples produced by the entropy source in this work belong to a fixed alphabet of two symbols - the output of the entropy source produces one bit at a time. This assumption eliminates the need for *Numerical predictors*, as they become identical to the predictors that already exist in *Categorical data predictors* set.

The *Categorical data predictors* set consists of four predictors. However, we noticed that the predictor *LZ78Y*, which gives prediction based on the most common sample that followed the current value according to the dictionary, shares concepts with *Markov Model with Counting* predictor and they both perform well with the entorpy sources that produce samples according to similar distributions [1]. The remaining three predictors are implemented and explained further in this section.

Each of the predictors is realized with several different model parameters, and all the models of one predictor are grouped together into one *ensemble predictor*. The ensemble predictor gives the prediction based on the success rate of its subpredictors. Therefore, our implementation consists of three ensemble predictors - prediction-based tests for min-entropy estimation. In order to be consistent with previous work [3], the tests in this letter also report eight levels of min-entropy on a scale from 0 to 7. Assessment of min-entropy level in all tests is done by calculating the percentage of successful predictions.

Most Common in Window (MCW) is a predictor that keeps track of the last w output samples of the entropy source, and bases its prediction on the most common sample observed. This predictor is designed for entropy sources that produce samples which have a clear most common value. *MCW* predictor is used as a part of ensemble predictor *MultiMCW*.

Single Lag Predictor gives as prediction the value equal to the output sample that appeared N observed samples back in the sequence. It is designed for entropy sources that produce sample sequences which express periodic behavior. This predictor is used as a part of ensemble predictor *MultiLag*.

Markov Model with Counting (MMC) predictor keeps counts for each sample that has followed each N-sample string. Whenever the string is observed again, this predictor gives a prediction equal to the most observed sample that followed

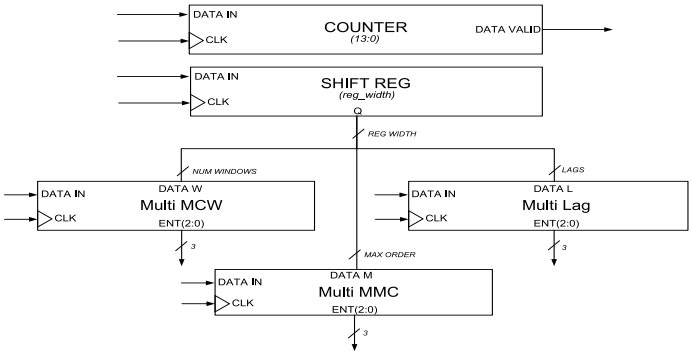

Fig. 1: General overview of the prediction-based test platform architecture.

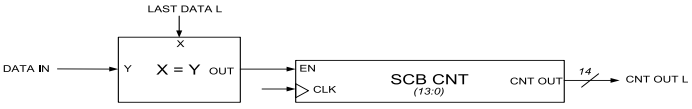

Fig. 2: Hardware architecture of the *Single Lag* predictor.

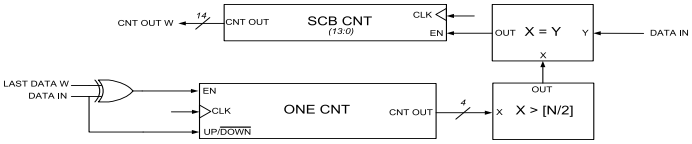

Fig. 3: Hardware architecture of the *MCW* predictor.

that string in the past. The *MMC* predictor is used as a part of ensemble predictor *MultiMMC*.

III. Implementation

An architectural overview of the implemented prediction-based test platform, which contains all prediction-based tests, is shown in Figure 1. All implemented tests use one common shift register to store bit samples produced by the entropy source. The width of this register corresponds to the maximum number of samples needed for one of the tests. The test platform also contains a 14-bit wide counter used solely for the detection of the end of the bit sequence.

Each predictor in one test contains a local *scoreboard* (SCB) counter which is updated every time the predictor makes a successful prediction. Each test contains a global *scoreboard* comparator to select the prediction given by the predictor which was most successful in the past; a global *scoreboard* counter which is used for min-entropy level assessment and a comparator with 7 predefined cut-off values for min-entropy level assessment of the bit sequence. After the end of each bit sequence, all *scoreboard* counters in each test are reset. The final min-entropy level of the bit sequence is computed as the minimum of all min-entropy levels given by each test.

Figure 2 shows the implementation of the *Single Lag* predictor which is a part of the *MultiLag* test. Due to its simplistic behavior, this predictor can be implemented with one shift register, one comparator and one counter.

Figure 3 shows the implementation of *MCW* predictor which is a part of the *MultiMCW* test. Since in our work the sample space of the entropy source has only two values

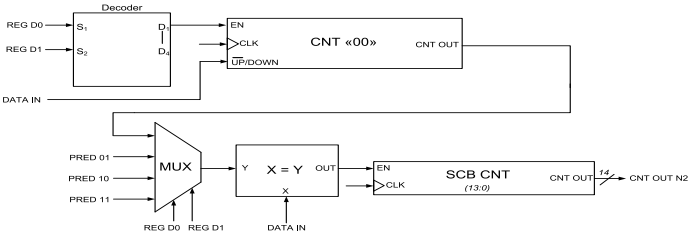

Fig. 4: Hardware architecture of the *MMC* predictor.

(0 and 1), we observe that this predictor's statistics can be described with only one parameter - number of 1s observed in the last w output samples. If this parameter is greater than $w/2$, the prediction of *MCW* is 1, otherwise it is 0. This approach enables us to implement the *MCW* predictor by using one shift register, two counters, two comparators and one XOR gate.

Figure 4 shows part of the implementation of *MMC* predictor. A straightforward implementation of the *MMC* predictor has a rather complex structure, but several simplifications can be made in order to make a lightweight version of this predictor. Namely, keeping track of the number of each sample that follows each string can be implemented with one up/down counter. The counter works in down mode whenever bit value 1 is observed, and in up mode whenever bit value 0 is observed. Thus, we can further reduce the number of needed comparators, since the first bit of the up/down counter can be directly used for comparison with the current output of the entropy source. Each predictor contains 2^N counters, where N is the length of a bit string, that are used for keeping track of the number of each sample that followed the string. Whenever the string is observed again, the counter is updated, and the most significant bit of the counter is selected by the multiplexer as the corresponding prediction of the *MMC* predictor.

IV. Selection of the Implementation Parameters

The parameter space of our test platform consists of bit sequence length and test specific parameters (window sizes for *MultiMCW*, lag depths for *MultiLag* and lengths of strings for *MultiMMC*). For the sake of comparison of different versions of tests according to their prediction performance, we establish two figures of merit - precision of predictions and accuracy of the predictions. Precision of predictions refers to variability of the test reported min-entropy for different bit strings of the same characteristics, whereas accuracy of predictions refers to proximity of the test reported min-entropy to 8-levels quantized theoretical min-entropy.

Precision of predictions is directly proportional to bit sequence length because the tests give more consistent predictions when they are provided with more data. Therefore, the length of the bit sequence is determined as a compromise between two contradictory requirements: the need for more reliable results (requiring long bit sequences) and the fast detection of failures of the entropy source (low latency). We initially started our validation experiments, described in Section V, by choosing the sequence length of 2^9 bits and recording the highest min-entropy span reported by the tests.

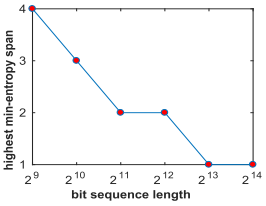

Fig. 5: Precision analysis

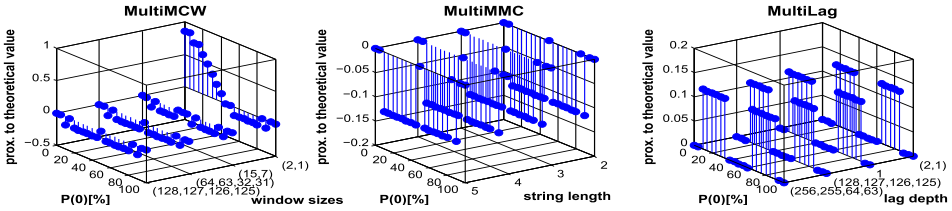

Fig. 6: Accuracy analysis for entropy source with varying bias

All three tests reported min-entropy levels of high variability, spreading across 4 levels for some bit sequences, independent on test specific parameters. Hence, we have preformed the same validation experiments for bit sequences of lengths 2^{10}, 2^{11}, 2^{12}, 2^{13} and 2^{14}, and recorded highest reported min-entropy spans depending on the bit sequence length. These results are shown in Figure 5. The shortest bit sequence length, for which all tests report min-entropy levels with variability not higher than 1 level, is 2^{13}. This level of variability is acceptable for the use in on-line testing, and thus is our starting point for accuracy analysis. Accuracy of predictions depends on bit sequence length, test specific parameters and suitability of the test for the entropy source (as described in Section II). The accuracy of the tests that are suitable for the tested entropy source increases with longer bit sequences, but the accuracy of those tests can also be improved by exploring a second degree of freedom - test specific parameters. The accuracy of prediction-based tests that are unsuitable for the tested entropy source does not increase with longer bit sequences, but it can be somehow enhanced by choosing adequate test specific parameters. Because of the complex dependency of accuracy on test specific parameters, we implemented different versions of each test in order to establish that dependency and find lightweight implementations that at the same time most correctly predict min-entropy of the entropy source. The accuracy analysis was performed on these tests by calculating the proximity of the highest reported min-entropy level to the corresponding 8-level quantized theoretical min-entropy of 2 entropy sources - unbiased generator with varying correlations and generator without correlations with varying bias. Figure 6 shows the results of accuracy analysis for several characteristic values of the test specific parameters when tests are performed on the data produced by entropy source with varying bias. It can be seen that for both *MultiMCW* and *MultiMMC* tests several different versions preform equally well, so we decide for the most lightweight versions - 7 and 15 for window sizes of *MultiMCW* and 1 and 2 samples for *MultiMMC* test. Because of the unsuitability of *MultiLag* test for this entropy source, all versions of this test report higher min-entropy level, so we decide to implement the one that has the lowest number of deviations - lags 2 and 1. Similar analysis was performed with entropy source with varying correlations, and analogous conclusion to these were made. In order to be sure about the level of achieved accuracy of our tests, we increased the length of the bit sequence to 2^{14} bits and repeated the experiment. This showed that approximately 1 level better accuracy can be achieved with this bit sequence for *MultiMCW* and *MultiMMC* tests. However, since 1 level of min-entropy variability is acceptable for the on-line testing, we decided to keep value 2^{13} as the length of the bit sequence in order to achieve lower latency and at the same time being sure that our estimations are no more than 1 level off.

V. Validation of Implemented Tests

The validity of the implemented prediction-based tests is verified by simulating the non-ideal entropy sources and comparing the results produced by the tests with the theoretical value of the min-entropy of the entropy source. Three types of non-ideal sources have been used: an unbiased generator with correlation between consecutive bits [3], a biased generator without correlation between consecutive bits [3] and a sampled ring oscillator. The theoretical value of the min-entropy of the entropy source is represented using continuous line in all figures of this section.

Figure 7 shows test results for the entropy source which produces unbiased bit sequences with correlation between consecutive bits. The bit flipping probability was varied from 0% to 100% in steps of 5%. For each step, 100 sequences of 2^{13} bits were generated. The graph named *Final* represents the final min-entropy estimate that is taken as the minimum of all tests. It can be seen that the only test that is able to accurately estimate the min-entropy of this entropy source is *MultiMMC*. As expected, the *MultiMCW* test fails because there is no clear common value in bit sequences whose bits randomly flip with certain probability, although no full min-entropy is reported until the probability of bit flipping reaches 50%. The *MultiLag* test is almost able to follow the theoretical min-entropy because it is able to predict the next output of the entropy source when the probability of bit flipping is very low, very high, or equal to 50%, since at this points certain periodicity of bit sequences can be observed.

Figure 8 shows test results for the entropy source which produces biased bit sequences. The bias in bit sequences is expressed by the probability of the bit being equal to 0. The probability of 0 was also varied from 0% to 100% in steps of 5%. For each step, 100 sequences of 2^{13} bits were generated. In this case, it can be seen that the *MultiMMC* test is again able to accurately estimate the min-entropy of the entropy source. The *MultiMCW* test also accurately estimates the min-entropy because there exists a clear common value in every bit sequence, which depends on the value of probability of appearing 0. It can be observed that the *MultiLag* test behaves in the same way as in the previous case, as this test is not designed to predict this type of bit sequence behavior.

As the final step in the verification process, the effectiveness of the proposed tests was evaluated using data generated by

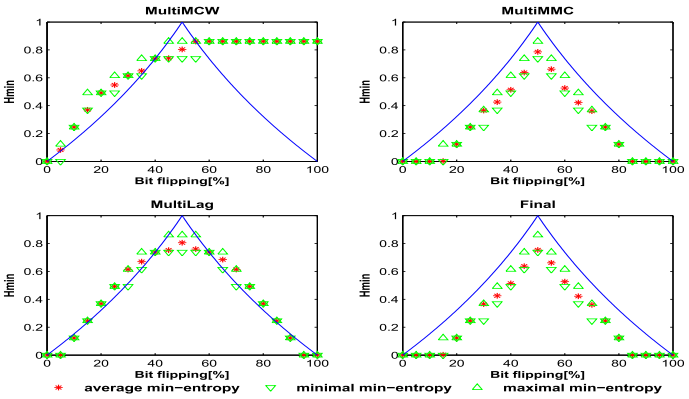

Fig. 7: Test results of the sequences produced by the unbiased generator.

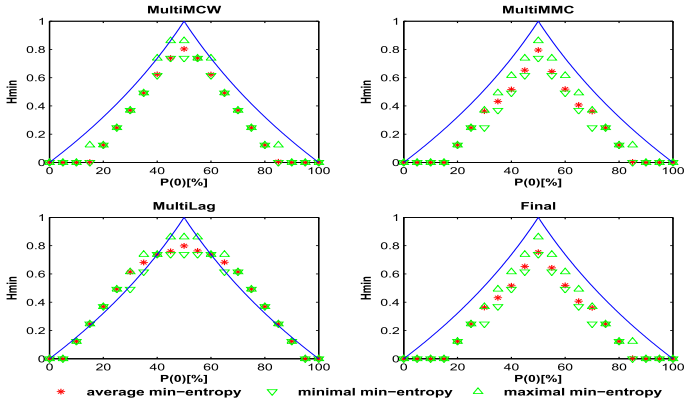

Fig. 8: Test results of the sequences produced by the biased generator.

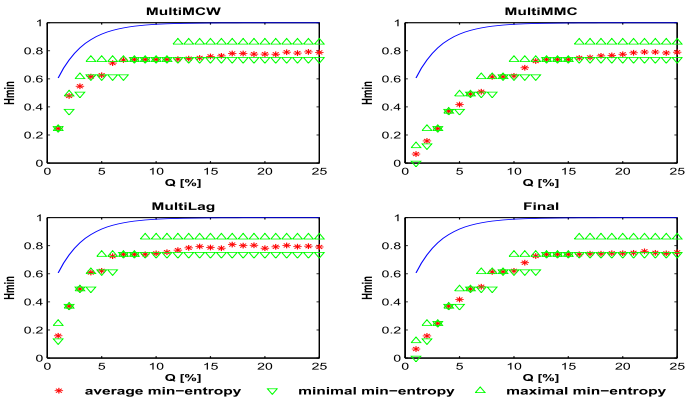

Fig. 9: Test results of the sequences produced by sampling ring oscillator.

a simulated ring oscillator. The data were generated using a Matlab model of an oscillator with white noise jitter. We varied the quality factor, which represents the phase variance accumulated between two samples, from 1% to 25% in steps of 1%. The results are presented in Figure 9. For each step, 100 sequences of 2^{13} bits were generated. The test results of all three implemented tests follow the same trend as theoretical min-entropy of the source, as presented in [4]. The observed trend can be properly explained by the fact that for very low values of the quality factor, which correspond to high sampling rates, this entropy source produces bit sequences with alternating runs of 0s and 1s. *MultiMCW* test is now able to determine a clear common value in sliding window, whereas the *MultiLag* test can detect periodic behavior in the runs of 0s and 1s. *MultiMMC* test can successfully estimate the min-entropy of all three sources since all three can be modeled by a Markov model.

VI. Conclusion

The proposed hardware design is synthesized using Xilinx ISE 14.7 on Spartan-6 XC6SLX45 FPGA. Modelsim SE 6.6d and ISim 14.1 are used for functional simulation. The implementation result of the test platform (unified implementation of all tests), with implementation parameters from Section IV, is given in Table I. Our design uses 87 slices and it can work on maximum frequency of 187.5 MHz. The separate implementation results of each test are also given in Table I. These results do not include logic for min-entropy level assessment, since it is the same size inside each test. It should be noted that every slice is counted in every test that has its basic elements packed in that slice.

TABLE I: Implementation Results of Tests

Test	Slices	LUTs	FFs
Unified implementation	87	282	223
MultiMCW test	18	50	54
MultiMMC test	38	134	132
MultiLag test	4	14	14

In this letter, we proposed a hardware implementation of prediction-based tests for min-entropy estimation of sources that produce non-ideal data. The achieved throughput of 187.5 Mbit/s and small resource consumption makes our design suitable for use with FPGA TRNGs in resource constrained environments where the security of the system is of paramount importance. From the conducted experiments we deduce that the *MultiMMC* test can be used to detect all types of failures presented. This is supported by the fact that a large number of physical processes, which are basis for randomness extraction in TRNGs, can be modeled with a Markov model. However, the *MultiMMC* test is at the same time the part of our design that consumes most hardware resources on FPGA, which can be clearly observed in Table I. The *MultiMCW* and the *MultiLag* tests are much smaller, but can be used to detect only limited number of failures.

References

[1] J. Kelsey, K. A. McKay, and M. S. Turan, "Predictive models for min-entropy estimation," in *Cryptographic Hardware and Embedded Systems - CHES 2015* (LNCS 9293), Berlin, Heidelberg, Germany: Springer-Verlag, Sep. 2015, pp. 373-392.

[2] B. Yang, V. Rožić, W. Dehaene, N. Mentens, and I. Verbauwhede, "TOTAL: TRNG On-the-fly Testing for Attack detection using Lightweight hardware," in *Proc. Design, Automation and Test in Europe (DATE)*, Dresden, Germany, Mar. 2016, pp. 127-132.

[3] B. Yang, V. Rožić, N. Mentens and I. Verbauwhede, "On-the-fly tests for non-ideal true random number generators," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, May 2015, pp. 2017-2020.

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