FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Hardware Generated Keys for Cryptographic Systems and Protocols

by

Simona Buchovecká

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Information Security

Prague, August 2020

Supervisor:
prof. Ing. Róbert Lórencz, CSc.
Department of Information Security
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

The main topic of this dissertation thesis is the generation of cryptographic keys in hardware and embedded systems.

For lightweight and embedded devices, the True Random Number Generators (TRNGs) are usually implemented, utilizing non-deterministic effects in analogue or digital circuits, since this is resource and power efficient way. In the dissertation thesis we propose and analyze the secure TRNG design, as well as we deal with the proper testing of hardware based TRNG, attempting also attacking the device.

Further, we present the authentication protocols based on Physically Unclonable Function (PUF) as the PUFs usage is promising to solve the issue of secure storage of cryptographic keys. Instead of storing the key in memory, the key is generated at the time it is needed. We designed combined PUF/TRNG circuit as a suitable alternative for the purpose of key generation and authentication. We show the possibilities of securing communication and authentication of the embedded systems and simple micro-controllers used in Internet of Things (IoT) devices, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and micro-controllers.

In particular, the main contributions of the dissertation thesis are as follows:

1. Proposal of TRNG design based on Ring Oscillator PUF (ROPUF) circuit, enabling the simultaneous generation of PUF and TRNG using the same hardware component, suitable even for simple micro-controllers and embedded devices

2. Proposal of the protocols for lightweight authentication and secure communication for IoT and embedded devices showing the possibilities of securing communication and authentication of the embedded systems, using PUF/TRNG combined circuit as a basic building block, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and micro-controllers.

3. Presentation of frequency injection attack variation, that allows to influence the generated sequence and decrease the randomness of generated bitstream, stressing out the importance and need of on-line testing.

Keywords:
cryptographic key, key generation, key storage, key management, TRNG, PUF

Acknowledgments

First of all, I would like to express my gratitude to my dissertation thesis supervisor, Prof. Róbert Lórencz. He has been a constant source of encouragement and insight during my research and helped me with numerous problems and professional advancements.

Special thanks go to my collaborators from Department of Information Security, Jiří Buček and Filip Kodýtek, for valuable contribution to my work.

Finally, my greatest thanks go to my partner and my family, for their infinite love, support and encouragements.

My research has been partially supported by the Grant Agency of the Czech Technical University in Prague, grants No. SGS14/107/OHK3/1T/18, SGS15/120/OHK3/1T/18 and SGS17/214/OHK3/3T/18.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AIS	Anwendungshinweise und Interpretationen (Instructions for use and interpretation)
ATM	Automated Teller Machine
BER	Bit Error Rate
CMOS	Complementary Metal Oxide Semiconductor
DRNG	Deterministic Random Number Generator
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
HSM	Hardware Security Module
HTTPS	Hypertext Transfer Protocol Secure
IC	Integrated Circuit
IoT	Internet of Things
LCD	Liquid-Crystal Display
LSB	Least Significant Bit
LSFR	Linear Feedback Shift Register
MSB	Most Significant Bit
NLFSR	Non-Linear Feedback Shift Register
NPTRNG	Non-Physical True Random Number Generator
PBKDF2	Password-Based Key Derivation Function 2
PTRNG	Physical True Random Number Generator
PUF	Physical Unclonable Function
RC Oscillator	Resistance-Capacitance Oscillator
RFID	Radio-Frequency IDentification
RO	Ring Oscillator
ROPUF	Ring Oscillator PUF
RNG	Random Number Generator
SRAM	Static Random Access Memory
SSH	Secure Shell
TRNG	True Random Number Generator

Chapter 1

Introduction

Cryptography has become an inevitable part of information technology, used to protect data that is sensitive, has a high value, or is vulnerable to unauthorized disclosure or undetected modification during transmission or while in storage. Cryptography relies upon two basic components: public algorithm (or cryptographic methodology) and a secret cryptographic key. The algorithm is a mathematical function, and the key is a parameter used by that function [5].

As stated in [31] the security of cryptographic systems is mainly linked to the protection of confidential keys. In high end information security systems, when used in an uncontrolled environment, cryptographic keys should never be generated outside the system and they should never leave the system in clear. For the same reason, if the security system is implemented in a single chip (cryptographic system- on-chip), the keys should be generated inside the same chip.

As discussed in [65], minimal common requirements for key generation and storage are:

1. A source of true randomness that ensures unpredictable and unique fresh keys,
2. A protected memory which reliably stores the key's information while shielding it completely from unauthorized parties.

However, these requirements are often neglected as they are not easy to be implement properly and generally non-trivial to achieve.

1.1 Problem Statement

Implementation of proper methods for cryptographic key generation and their secure storage in embedded devices (including programmable logic devices) is of significant importance. Moreover, it is necessary to use the cryptography and corresponding keys in proper manner.

Various papers presented successful attacks on embedded systems, due to improperly implemented key generation and key management. Nowadays, TRNGs are mostly used for key generation. Therefore, the quality of random number generator has a significant influence on security of whole cryptosystem. Improperly implemented RNG often leads to compromise of the whole system or reduces the complexity of the attack as discussed in next paragprahs.

The attack on Mifare Classic tags [80] targets and recovers smart card's secret key, taking advantage of highly insecure RNG used for key generation. The attacker was allowed to compute a code book that decreased the complexity of the attack dramatically.

Sony PS3 protection was broken, because of parameter that should have been random and unique for each ECDSA signature computation, was not randomized properly; thus, allowed attackers to compute private key [42].

Defective random number generators implemented in various ATMs and Point-of-Sale terminals, that are often just counters, allows attackers to harvest authentication codes which enable a "clone" of the card to be used in ATMs and elsewhere [11].

The flaws in design of key generation and storage are described in [18] – the authors analyze more than 4000 embedded devices of over 70 vendors and found out, that keys "have been embedded, essentially "baked in" the firmware image (operating system) of devices and are mostly used for providing HTTPS and SSH access to the device. This is a problem because all devices that use the firmware use the exact same keys."

Moreover, once key is generated, it needs to be stored in stored securely [38] e.g. utilizing storage with tamper-resistance techniques implemented. However, to implement such measures is complex and cost-ineffective task, therefore often neglected in practical applications.

With the evolving usage of embedded systems and boom of IoT, the issue of proper generation, storage and management of cryptographic keys in hardware devices is growing its importance. Thus the need for securing the communication is increasing. Cryptographic protocols rely on security and quality of the key, however, there is no clear and unified methodology how to manage overall life cycle of hardware generated keys within hardware devices and embedded systems.

The simpler the device is the more difficult implementation of the cryptographic protocols (including those for secure key handling) is, as the implementation of the cryptographic primitives is resource exhaustive. Thus, efficient key management suitable for simple embedded devices, including proper key generation, key storage and key usage for various application in embedded systems, including applications in IoT, is necessary. However, this issue is not properly addressed nowadays. Most of the papers presented in this area only deal with process of generating TRNGs and PUFs (that are suitable for cryptographic applications), but usually do not elaborate further the proper key management, storage and secure usage.

1.2 Structure of the Dissertation Thesis

The thesis is organized into five chapters as follows:

1. *Introduction*: Describes the motivation behind our efforts together with our goals.

2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art. The problematics of cryptographic key generation is introduced, including the primitives - TRNGs and PUFs that are being used for key generation, as well as the protocols used for secure authentication and communication of hardware devices.

3. *Overview of Our Approach*: Discusses our approach to the TRNG design and it's testing and further the ideas leading to the protocol for secure authentication and communication that is enabled by single combined TRNG/PUF circuit for high efficiency in simple and embedded devices.

4. *Main Results*: Presents our main results in the form of a collection of reviewed and published relevant papers.

5. *Conclusions*: Summarizes the results of our research, suggests possible topics for further research, and concludes the thesis.

CHAPTER 2

Background and State-of-the-Art

The security in last years has been subject of intensive research, however, despite this efforts, it is still often misconstrued by designers of hardware devices. At the same time, the interconnected embedded systems within IoT are gaining on importance and multiple authors are researching the area, identifying the current issues and formulating requirements. Embedded system security requirements are discussed in the paper [55] and authors further formulate the security functions should be always provided as minimum: data integrity, data confidentiality, user identification and authentication, secure communication, secure network access, secure content and its storage and tamper resistance. Similarly, in [48] authors discuss the need for systematic and proactive approach to security and privacy in IoT and identify key challenges in security: data provenance and integrity, identity management, trust management and privacy.

Systematic and formalized approach to key management in hardware devices and embedded systems with properly defined requirements, as well as efficient light-weight modules for key generation and storage and secure usage are missing. However, the need for proper key management in particular applications of embedded systems and IoT started to being raised in some of research papers - the various aspects of security and privacy within simple hardware devices interconnected in IoT are discussed e.g. in [16, 71, 90, 100, 103].

Assessment of the performance of various cryptographic primitives on various micro-controllers, smart-cards and mobile devices is presented in [71]. While the authors show that there are schemes that are light-weight and suitable for even simple and constrained devices, they state that there is need for tamper-proof modules when using such schemes.

The need for built-in security and countermeasures against successful breaches is also discussed in [103], presenting research on current research challenges of IoT. One of the key areas according to the paper is authentication, confidentiality and access controls. Authors conclude the open issues that need to be resolved that includes how to handle different keys, if is it possible to reuse traditional security mechanisms or how to ensure end-to-end integrity verification mechanism.

In [100] authors discuss the requirements and give best practice approaches for a secure key management solution in the automotive context. The security vulnerabilities as well as best practice approaches for secure key management are presented both in the embedded in-vehicle domain as well as for supporting back end infrastructure.

Overview of a key management problem for distributed sensor networks is introduced in [16]. The paper presents a precise formulation of the distributed revocation problem as well as an initial protocol that has been shown to satisfy the requirements of this problem formulation.

Finally, as summarized in [90], all security protocols require credentials, thus it is obvious the optimal key management systems have to be implemented to store and distribute these credentials as the necessary prerequisite.

2.1 Cryptographic Key Generation

The generation of cryptographic keys is the first and essential step in the key life cycle. The generated key needs to meet the strict requirement of its unpredictability, arising from Kerckhoffs' principle formulated by Auguste Kerckhoffs in 1883 [51]. The principle states that the cryptographic system should be secure even if everything about the system, except the key, is public knowledge. This principle is applied to all modern encryption cryptosystems and the algorithms for encryption are publicly known. Therefore, the key needs to be kept secret and unpredictable, so the attacker cannot easily guess it. In hardware, the Random Number Gerenators (RNGs) or Physical Unclonable Functions are used to generate unpredictable bitstream. Further, postprocessing of this bitstream allows to generate the cryptographic key, as discussed in following sections.

2.2 Random Number Generators

As discussed in previous chapter, one of the main requirement on the cryptographic keys is their unpredictability and randomness, thus the RNGs are often utilized for the key generation.

A RNG can be defined as a device or algorithm which outputs a sequence of random (thus independent and uniformly distributed) numbers. In practical hardware implementations, the output sequence is represented as a bit stream of zeros and ones, that may be further sliced and converted to the integers, as per the need of the implemented algorithms.

According to the [99], the RNGs can be classified following the scheme depicted in the Fig. 2.1.

The first group contains deterministic RNGs (DRNGs, also pseudo-random number generators). DRNGs generate the numbers algorithmically and need the so-called seed as an input, which determines the run of the algorithm and generated sequence. The selection of the seed is crucial because one seed always generates the same sequence.

The true RNGs (TRNGs), belong to the second group, and form two sub-groups; physical TRNGs (PTRNGs) and non-physical TRNGs (NPTRNGs). Physical TRNGs use non-deterministic effects of electronic circuits (e.g., shot noise form Zener diode, inherent semiconductor thermal noise, free-running oscillators) or physical experiments (e.g. time between emissions of radioactive decay, quantum random processes). NPTRNGs exploit non-deterministic events such as system time, hard disk seek time, RAM content, user interaction. So-called hybrid RNGs have design elements from both DRNGs and TRNGs. The security of DRNGs essentially depend on the computational complexity of their output (practical security), while TRNGs rely on the unpredictability of their output (theoretical security). Depending on their main ”security anchor” it is distinguished between hybrid DRNGs and hybrid TRNGs.

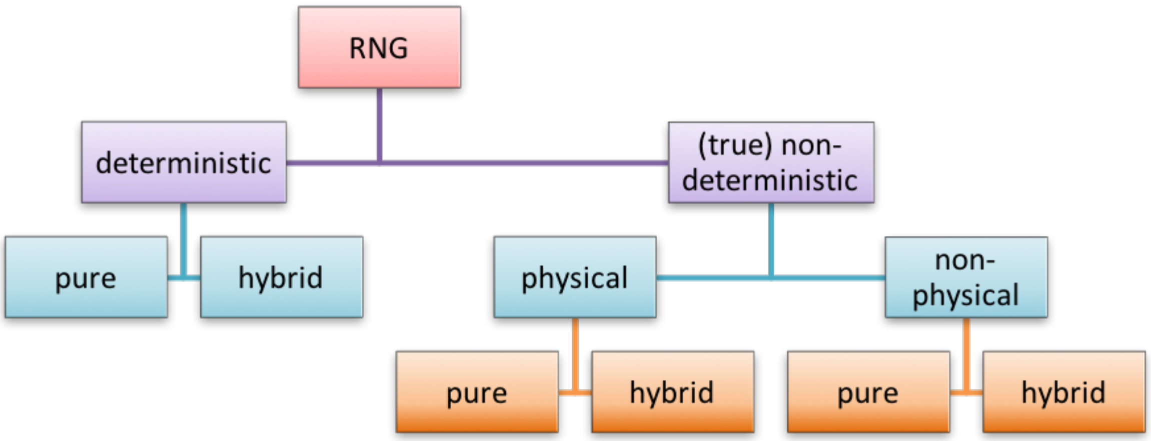


Figure 2.1: RNG classification as per [99].

The DRNGs are usually significantly resource exhaustive as there purely algorithmically based and, moreover, there is a strong need for an unpredictable and fresh seed. Thus, when dealing with physical embedded devices the TRNG is often the first choice, as there are multiple sources of the entropy usually available on the device itself (such as non-deterministic effects of electronic circuits or various analog sensors present).

The generic design of PTRNG is described in [99] and depicted in Fig. 2.2. The basic building block is the noise source (thus, source of the entropy) typically realized by electronic circuits (e.g., using noisy diodes or free-running oscillators) or by analog sensor (e.g. measuring the noise level or evaluating visual data). The noise source generates time-continuous analog signals which are digitized to binary values (digitized analog signals or briefly das random number numbers). The das random numbers may be algorithmically post-processed to internal random numbers in order to reduce potential weaknesses. The reduction of weakness simply mean we transform it into other from - e.g., bias into dependencies, and requires data compression which in turn lowers the output rate of the RNG. The algorithmic post-processing may be memory-less (depending only on the current das bit), or it may combine the current das random numbers with memory values that depend on the preceding das random numbers (and possibly some other tuning parameters).

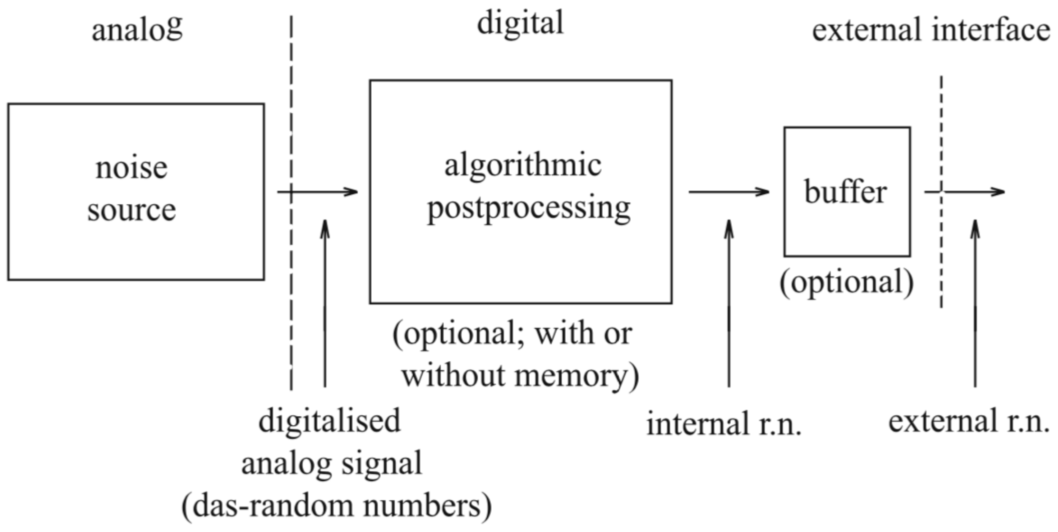

analog
digital
external interface
noise source
algorithmic postprocessing
buffer
(optional)
(optional; with or without memory)
digitalised analog signal (das-random numbers)
internal r.n.
external r.n.

Figure 2.2: Generic design of a PTRNG as per [99].

2.2.1 Evaluation of Generated Sequence

The security of the TRNGs relies on the unpredictability of the generated sequence. The necessary requirements are summarized in [98, 99]:

- R1 -The random numbers should have good statistical properties.
- R2- The knowledge of sub-sequences of random numbers shall not allow one to practically compute predecessors or successors or to guess these numbers with non-negligibly larger probability than without knowledge of these sub-sequences.
- R3 - The knowledge of the internal state shall not allow one to practically compute 'old' random numbers or even a previous internal state or to guess these values with non-negligibly larger probability than without knowledge of the internal state.
- R4 - Even the knowledge of the internal state shall not allow one to practically compute the next random numbers or to guess these values with non-negligibly larger probability than without the knowledge of the internal state.

The "good statistical properties", as per R1, we understand the generated sequence has uniform distribution and the individual generated bits are independent, thus we can represent the generated bit sequence as a random variable:

$$P\{X = x_i\},\ i = 1, 2, ..., n,\ x_i \in \{0, 1\} \tag{2.1}$$

following the discrete uniform distribution on the values in the sequence $x_i, ..., x_{i+n}$, with probability mass function:

$$f(x) = \frac{1}{n}. \tag{2.2}$$

However, the requirement of uniformity and independence is not sufficient itself (for instance, Linear Congruential Generator will fulfill the R1, however, it is not suitable for cryptographic use, as the next generated bit can be easily guessed). Therefore, the additional requirement R2 calls for unpredictability of the generated bits - which is closely related to the entropy of generated sequence. Further, in case of TRNGs, the requirements R3 and R4 usually follow immediately from R2 in majority of the implementations and since the TRNGs rely solely on the unpredictability of the generated random sequence.

Measuring of uncertainty is closely related to the concept of information entropy introduced by Claude Shanon in [102]. Shanon based his work on Nyquist's and Hartley's papers [39, 81] and defined the information entropy as follows:

$$H(X) = -\sum_i p_i \; log_2(p_i), \tag{2.3}$$

or in alternative notation:

$$H(X) = \sum_x Pr[X = x] \; log(Pr[X = x]), \tag{2.4}$$

where X is a random variable with probability distribution p_i.

However, since we need to achieve unpredictability and good statistical properties of the generated sequence, neither the entropy measurement itself is not enough to evaluate the quality of the RNG at its own.

Practically, the generated sequence should be unbiased and uniformly distributed (each generated bit should have equal 50% probability for 0 and 1) and each generated element should be independent from its predecessors (allowing forward and backward unpredictability).

2.2.2 Evaluation and Testing of TRNGs

During TRNG design phase, we need to ensure the quality of the generation itself. Various publications dealing with matter of testing RNGs (both TRNG and PRNG) exist. Most of them provide general advice and are not specific for cryptography, such as Knuth's Art of programming Semi numerical algorithms [53] or Marsaglias Diehard Battery of Tests of Randomness [72]. Requirements on TRNGs and their evaluation is further discussed in more details in [98, 99, 107]. For testing RNGs used specifically for cryptographic applications NISTs special publication - "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications" [93], accompanied by battery of statistical tests, is most often followed, and is perceived as standard suite of tests, as it was created with security applications of RNGs in mind and focuses on testing statistical qualities of the whole sequence, as well as selected sub-blocks of the sequence. The suite consist of 15 tests:

- **Frequency Monobit Test**, determining whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence, assessing closeness to fraction of ones to 1/2.
- **Frequency Test within a Block**, assessing the proportion of ones within M-bit blocks.
- **Runs Test**, identifying total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits, determining whether the oscillation between zeros and ones is too fast or too slow.
- **Test for the Longext Run of Ones in a Block**, determining the longest run of ones within M-bit blocks.
- **Binary Matrix Rank Test**, the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence. This test also appears in the DIEHARD battery of tests [72].
- **Discrete Fourier Transform (Spectral) Test**, testing peak heights in the Discrete Fourier Transform of the sequence. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.
- **Non-overlapping Template Matching test**, identifying the number of occurrences of pre-specified target strings. The purpose of this test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern, an m-bit window is used to search for a specific m-bit pattern.
- **Overlapping Template Matching Test**, similar as above test looking for predefined strings. The difference between this test and the test above is that when the pattern is found, the window slides only one bit before resuming the search.
- **Maurer's Universal Statistical Test**, assessing number of bits between matching patterns (a measure that is related to the length of a compressed sequence). A significantly compressible sequence is considered to be non-random.
- **Linear Complexity Test**, testing the length of a Linear Feedback Shift Register (LFSR). The purpose of this test is to determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized by longer LFSRs.

- **Serial Test**, assessing the frequency of all possible overlapping m-bit patterns across the entire sequence.
- **Approximate Entropy Test**, similarly as above, the focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence.
- **CUmulative SUMs (CUSUM) Test**, focusing on the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence.
- **Random Excursions Test**, calculating the number of cycles having exactly K visits in a cumulative sum random walk.
- **Random Excursions Variant Test**, calculating the total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk.

2.2.3 On-line Testing of the Generated Sequence

Apart from standard tests that are used to evaluate the quality of TRNG in design phase, additional tests are needed to ensure the generated bit stream fulfills the required conditions all the time when TRNG is in operation. As stated in [98], tolerances of components of the noise source of aging aspects may affect the quality of generated bit stream and in worst case the entropy source may totally break down causing the generated bits to be constant. [98] further discuss the tests that are needed to ensure desired qualities of TRNG start-up test (to verify the principle functionality of the noise source when TRNG has been started), on-line tests (detect if the quality of the random numbers is sufficient) and so called tot tests for detecting total failure of the noise source. Several works were presented to address this issue [95, 96] implemented four of FIPS 140-2 [15] statistical tests in the same chip as TRNG. In [117] authors presented efficient hardware implementations of 8 NIST tests suitable for on-line monitoring of TRNGs.

2.2.4 Design Specific Tests

When testing a hardware RNG, standard test suites are not always sufficient to reveal the flaws in design. The introduction and few concepts of TRNG testing are discussed in [19] - As summarized by the author:

- If the TRNG pass statistical tests it does not necessarily mean it is ”good”,
- The TRNG tests should be tailored appropriately to the design of the TRNG,
- With knowledge of the TRNG design, we should concentrate on the spots, where the problems are most likely to arise.

In [19] the author further summarizes the potential deviations from uniform randomness that can occur in the generated sequence:

- Biased sequence, moreover, the bias may drift over the time,
- Correlated adjacent bits,
- External frequencies picked up - e.g. external electrical interference,
- Semi-conductor noise influence - low frequency noise.

These flaws in generated sequnce, however, may be removed by post-processing of the generated sequnce, as discussed in the following sections.

The problem of testing a hardware TRNGs was closely studied by Werner Schindler and Wolfgang Killmann, formalized in AIS 20/31 standard [52]. Similarly as above, authors emphasize the need to not only perform statistical "black-box" tests, but also understand the nature of the random source to rate the randomness of number generation.

As an example, there was a significant effort around oscillator-based RNGs, as they are very often used for true random number generation in embedded devices and hardware in general, e.g.:

- In [32] authors focus on precise method of jitter measurement (which is a source of the entropy in oscillator-based RNGs) that together with a selected statistical model can be used for entropy estimation fully matching the generator's principle,
- In [7] authors propose a way how to evaluate and control parameters of an RO, including its entropy rate and the biases of certain bit patterns,
- In [61] authors present a stochastic model to evaluate the entropy of oscillator-based TRNGs, and then deduce the requirement of design parameters (including the sampling interval) for sufficient entropy per random bit - showing the need to accustom to the original design and source of the entropy,
- Another aspect - characteristics of the oscillators in frequency domain or time domain terms, was presented in [79], poviding insight into the oscillator noise physics.

2.2.5 True Random Number Generators Designs

TRNG design is vivid topic among researches, and various different designs were introduced during the course of the time. The first design question is, what will be the suitable source of randomness. When talking about TRNGs, one of the most obvious choices are analog blocks or sensors. Another option is, especially in case of programmable devices, to take advantage of the time domain instabilities- metastabilities of logic circuits.

Various physical phenomenas in analog components can be utilized as the source of the randomness, examples include noise generated but the circuit itself(thermal noise, shot noise, flicker noise, avalanche noise) noise coming from the environment (atmospheric noise), or instabilities in the circuit (Resistance-Capacitance (RC) oscillator instability)

The [126] introduces the purely PTRNG design based on cascaded CMOS amplifiers. Cascading the simple amplifiers provides a significant noise signal, but it is neither white nor in normal distribution. Therefore, comparator is used to further compare the noise with a reference voltage. If the noise is greater than the reference, the comparator gives out a “1”; otherwise a “0” are produced. Authors state that if the reference is the mean value of the noise, the output bit stream will contain the same amount of “0” and “1”, and can prove that the “0”s and “1”s distribute independently no matter how the noise distribute.

The Intel RNG [44] uses a random source that is derived from two free-running oscillators, one fast and one much slower. The thermal noise source is used to modulate the frequency of the slower clock. The variable, noise-modulated slower clock is used to trigger measurements of the fast clock. Drift between the two clocks thus provides the source of random binary digits. The overall block diagram is depicted in Fig. 2.3.

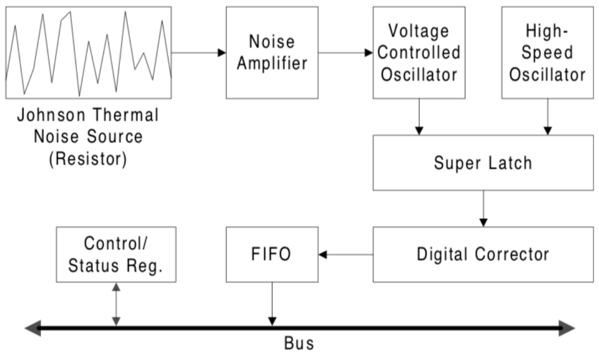


Figure 2.3: The Intel RNG design [44].

In [86], the authors built their RNG by implementing non-uniform quantization of avalanche diode noise samples. The RNG consists of both analogue part (avalanche diode and low noise amplifier), and sample processor, as depicted in Fig. 2.4. The main role of the sample processor is to perform non-uniform quantization that converts from Gaussian to uniformly distributed random process.

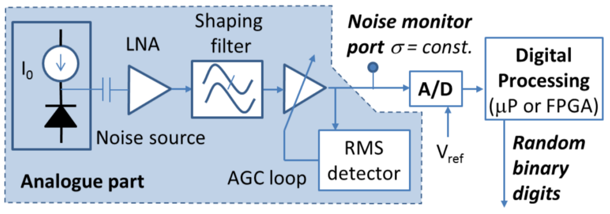


Figure 2.4: Peric RNG design [86].

However, in purely digital system design, especially when talking about programmable devices, there is usually no possibility to employ analog blocks and interact directly with them to get the noise signal directly. Thus, different sources of entropy have to be find. One of the options is to use metastabilities in digital circuits as possible source of entropy as presented in [8] or [29]. Bellido et. al proposed simple design using conventional binary latches, thus very suitable for implementation in any digital CMOS design, in [8], depicted in Fig. 2.5. Proposed design interconnects two R-S latches into cascade; the basic idea is to provoke a metastable state in latch L1 and read its final state using latch L2. Switches are used to remove the charge stored in the path connecting both flip flops. The caveat in here is impact of voltage changes on the generated sequence.

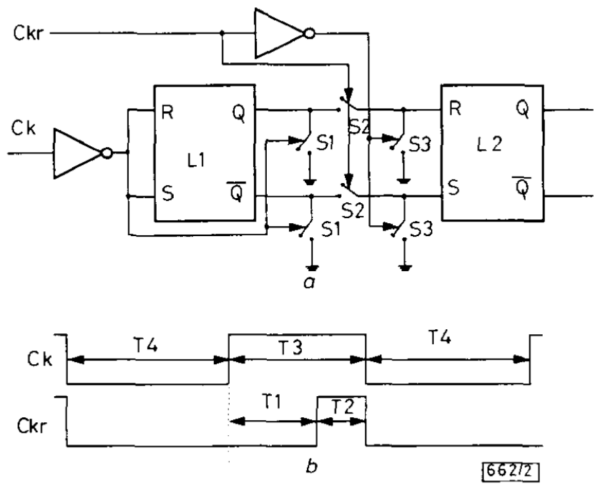


Figure 2.5: Bellido et. al design and timing of the circuit [8].

The design presented in [29], consisting of two inverters and four switches, behaves as bi-stable memory element, as depicted in Fig. 2.6. The authors discussed and tested several varieties of inverters to achieve different delays. Short (and different) delays are preferred because the ROs will produce smaller amplitude, sinusoid like signals, which should provoke metastability more often. Very short delays might prevent oscillation if the gain of the circuit is too small at the fundamental frequency of the feedback loop.

Another popular source of entropy in digital circuits are ROs, the source of the randomness being variations in the phase and frequency of free running oscillators (jitter). A RO consists of an odd number of logic inverters connected cyclically to form a ring. Typically, a high-frequency RO is sampled at a much lower speed by an independent (system) clock through a D-type flip-flop. If the sampling clock is generated by another RO, then there is a tendency of the ROs to couple with each other, thus significantly reducing the amount of randomness produced. [23]

Jitter generated in ROs is analyzed in detail in [112]. Each inverter of the RO propagates rising and falling edge of the generated clock signal in two half-periods respectively. The total period is thus given as $T = 2kd$, where $k \in 3, 5, 7, ..., 2n-1$ is the number of inverters and d is delay of one inverter. The delay instability of logic gates (inverters) connected

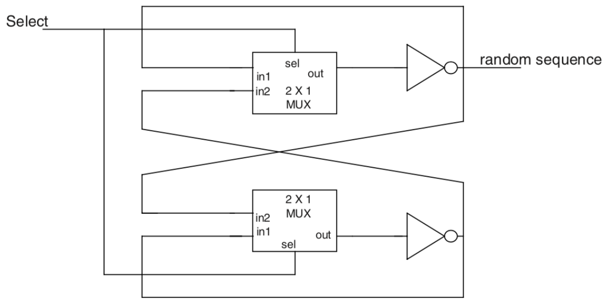


Figure 2.6: Epstein design based on bi-stable memory element[29].

into the ring is seen as frequency/phase instability (a jitter) of the generated clock. The jitter generated in ROs is composed of a random Gaussian jitter and a deterministic jitter. Authors point out the importance of evaluation of the proportion of the random jitter and deterministic jitter in the total clock jitter and their contribution to the generation of random numbers. It is especially important in the case of the deterministic jitter, because it can be manipulated from outside the device and it can thus constitute some attack on the generator. As further authors state, usually, the jitter is measured outside the device using fast digital oscilloscope. However, the input-output circuitry generates an additional jitter and the measured values do not correspond exactly to the jitter, which is used to generate random numbers. When modeling the jitter authors consider two components:

- **Local jitter sources** - in-deterministic jitter of the logic gate and local cross-talks from the neighboring circuitry,
- **Global jitter sources** - variation of the power supply and/or temperature, power source noise and some deterministic signal, which can be superposed on the supply voltage.

Moreover, we need to consider jitter accumulation in time and jitter added in routing and output circuitry.

In [35] Golic discussed two practical approaches to generating true random numbers that can be viewed as generalization of the RO structure and are based on a cascade of inverters and are replacing the simple circular feedback defining a RO by a more complex feedback incorporating a number of XOR logic gates in a way corresponding to the well-known Fibonacci or Galois configuration of an LFSR (Fig. 2.7). The inverters are used instead of the delay elements. Fibonacci RO consists of inverters connected in a cascade so that the output of each but the last inverter is directly used as the input to the next inverter. The feedback connections are specified by the binary coefficients fi (which are usually represented by a binary polynomial for both Fibonacci and Galois RO) with the convention that the corresponding switch is closed if $f_i = 1$ and open if $f_i = 0$, in which case the corresponding 2-input XOR gate is not present. The output signal could be taken from any inverter in the cascade. On the other hand, Galois RO consist of number of inverters connected in a cascade so that the output of each but the last inverter is used to form the input to the next inverter and the output of the output of the last inverter directly defines the feedback signal. The feedback connections are specified by the binary coefficients fi with the convention that the corresponding switch is closed if $f_i = 1$ and open if $f_i = 0$. If $f_i = 0$, then the input to the$i$-th inverter is directly defined by the output of the $(i+1)$-th inverter. If $f_i = 1$, then the input to the i-th inverter is formed by XORing the output of the $(i + 1)$-th inverter with the feedback signal.

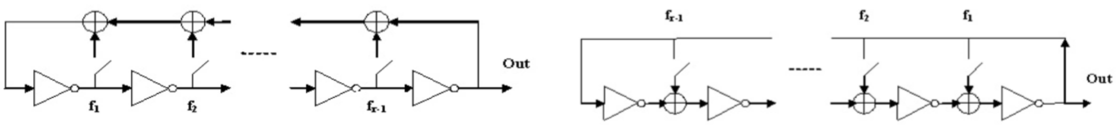


Figure 2.7: Fibonacci and Galois configuration of an LFSR [35].

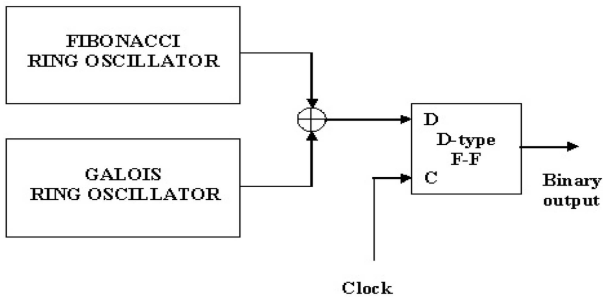


Figure 2.8: Golic's robust design [35].

Later, it was shown in [22] that there is a considerable risk of Fibonacci ROs to oscillate periodically. If this happens, the entropy produced is only a small percentage of the one from a Fibonacci RO working correctly. So, the failure of the Fibonacci ROs means a considerable risk for security applications needing reliable random numbers. This problem, however, may impact Galois ROs in some cases as well.

The randomness as well as robustness can be further increased by XOR-ing the outputs of two oscillators, one being in the Galois and the other in the Fibonacci configuration, as depicted in Fig. 2.8. Fibonacci ROs (FIRO) and Galois ROs (GARO) are also evaluated in [23], showing much better performance than classical ROs. The idea is to keep the FIRO and GARO in static reset state, allowing the oscillator to run for short period of time only when a random bit is needed. After sampling, the oscillator is stopped and reset to its initial state. A D-type flip-flop used for sampling should also be reset to a fixed state.

Another design combining multiple ROs was proposed in [108]. The design consists of free-running oscillators which outputs are XORed and sampled. The original design used 114 rings, each consisting of 13 inverters, the idea was further verified in [97] and the number of rings needed was refined to 110, each consisting of 3 inverters (Fig. 2.9). This design was further analyzed and elaborated in [23] and [122]. As per [23], the original design has several flaws - unrealistic probabilistic model of jitter, interaction amongst the ROs and unrealistic speed. Enhancement proposed in [122] adds an extra D flip-flop after each ring, as depicted in Fig. 2.10. The randomness of the configuration relies on the jitter variations of the oscillator rings. Adding these extra flip-flops will not alter the collection of the randomness of each ring, but improve the overall randomness at the output - the signals on the input of the XOR will now be synchronous with the sampling clock and only updated once in the sampling period. Due to this reduction in transitions on the input to the XOR tree, the setup- and hold-times for the internal logic in the FPGA will be within acceptable limits.

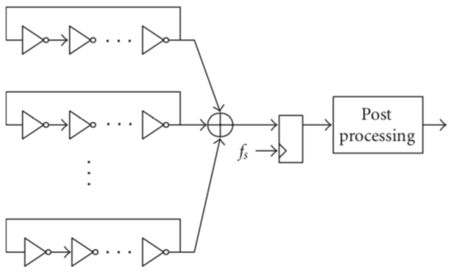


Figure 2.9: Parallel free running oscillators ring design - [108, 97].

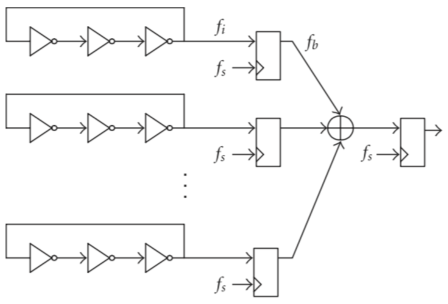


Figure 2.10: Enhanced free running oscillator design - [122].

Non-linear feedback ROs [110] are another variant of RO TRNG design. Since the Non-Linear Feedback Shift Registers (NLFSR) are considered to be better suited for cryptographic applications as they are more resilient and resistant to cryptanalysis than LSFRs, authors assume their suitability for random number generation. Design based on NLFSRs creates true randomness in both the initial condition and processing delay at each stage.

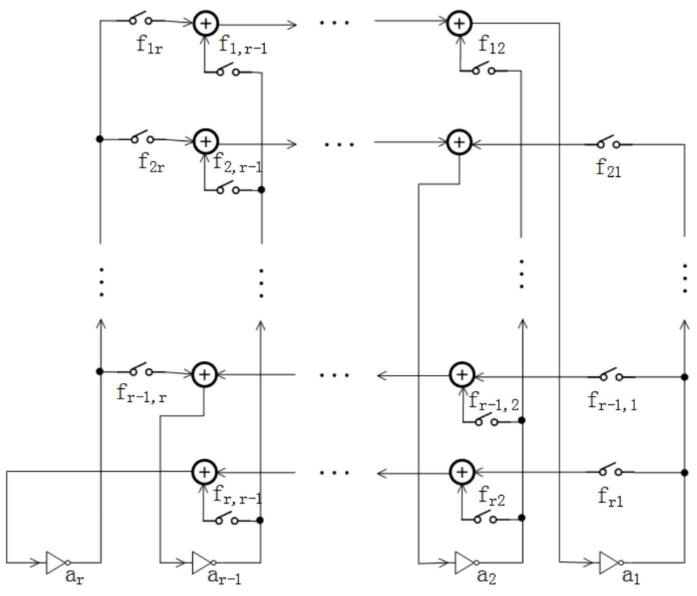


Figure 2.11: MFRO desgin - [110].

To overcome potential risks of FIRO/GARO oscillating periodically, matrix feedback RO (MFRO) was proposed in [124]. MFRO can be regarded as a generalization of FIRO/GARO designs. As depicted in Fig. 2.11, it consists of r inverters and some XOR gates. Its feedback loop is determined by two-dimensional matrix, the switches do not change their state during the operation. When there only exists one single feedback into inverter a_r, MRFO degrade into FIRO. When the input of feedback loop only comes from inverter a_1, MRFO become GARO. When there is only one XOR gate as the feedback input into an intermediate inverter, MFRO degrades to Boolean chaotic oscillator. Therefore the studies on how these related circuits generates chaos can be unified into a research on MFRO.

Edge Sampling TRNG (ES-TRNG) architecture based on high-precision edge sampling was introduced in [123]. The randomness source of the ES-TRNG is the timing phase jitter from a free-running RO. Two novel techniques are used to improve the throughput and reduce the resource consumption. The first technique is called variable-precision phase encoding. By using the selective high-precision sampling process, this technique enables a compact implementation and a short jitter accumulation time. The second technique is repetitive sampling, which allows multiple sampling within a single system clock cycle. Due to the repetitive sampling, ES-TRNG can obtain a higher throughput. By using a fully digital architecture and not relying on any technology specific components, we obtain a design feasible on a wide range of implementation platforms.

While most of the designs for generating random numbers on ROs are based on jitter that was questioned multiple times as mentioned above, in [116] rather the metastability phenomena in the RO for entropy accumulating is used. For practical realization of this method, a special RO architecture with the ability to be set in metastable mode was proposed - MetaRO based on odd plurality of inverters, corresponding number of MUXes (as switching components), control clock generator, sampling component (D flip-flop) and delay component. Meta-RO5st (5-stage metastable RO) component was implemented to validate the proposed model.

Another structure benefiting from metastabilities - Transition Effect RO (TERO) was introduced in [114, 115]. The basic structure of TERO block (Fig. 2.12) consists of even number of inverting elements and even number of XOR gates. The first input of XOR is employed in RO chain and the second (control input) is used for switching XOR from inverting to non-inverting mode. Such a circuit has two stable states due to positive feedback; the first, when XORs are configured as inverters and the second, when XORs are configured as buffers. When an edge appears in control signal, the RO is switched from one stable state to the second. This switching causes the transition effect - RO begins to oscillate for a while.

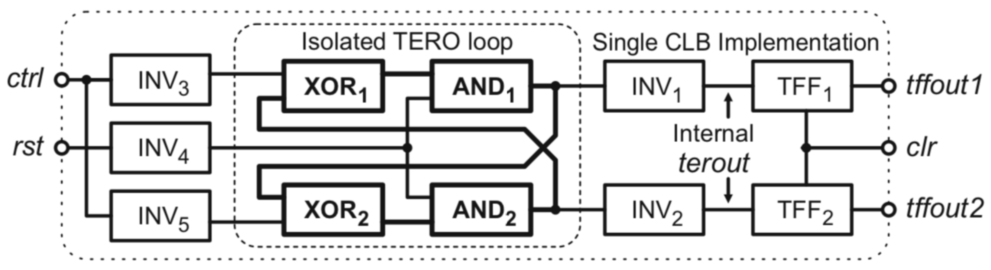


Figure 2.12: TERO structure - [114, 115].

Oscillator based RNG design is even suitable to be integrated in a Smart Card microcontroller [14] as well as other lightweight designs, as presented in [121]. For this purpose, a special TRNG D-Latch architecture (TDL) has been proposed, which can either operate in the ring-oscillator mode or the nearly-metastable mode.

2.2.6 Post-Processing of Generated Sequence

The generated random sequence may be algorithmically posprocessed to enhance statistical qualities and to increase the robustness of bias, mainly by reducing the bias and dependencies of generated output bits. The post-processing mechanism has to be carefully chosen, in order not to transform one flaw of generated sequence into another - e.g. bias into dependency. As such, good post-processing mechanism reduces the output rate of generated bit-stream.

Multiple post-processing techniques are employed in proposed designs:

- **Von Neumann Corrector** - simple technique introduced by Von Neumann in [119] working as follows: if the input is "00" or "11", the input is discarded, if the input is "10", output a "1", if the input is "01", output a "0". The technique is very effective producing uniform distribution of 0s and 1s ,unfortunately, the corrector shortens the output stream by 75%.
- **XOR Corrector** - takes two subsequent bits from input stream, XORs them and puts the resulting bit into the output stream, thus achieving better output rate than Von Neumann - only shortening the output by 50 %.
- **One-Way Hash Functions** - since the hash functions are one-way and non-linear, they can be a backup measure for the situations when the source of the entropy breakdowns (e.g. due to the attack), effectively transforming TRNG to PRNG. However, the implementation of the hash function consumes resources significantly.
- **Extractor Functions** - the concept of extractor functions introduced in [4], producing a result that is statistically close to the uniform distribution.
- **Resilient Functions** - as discussed in [108], seeking to eliminate these non-random deterministic components.
- Extractor functions based on linear or cyclic codes, such as **BCH Code** [85].
- **Chaos based post processing** introduced in [2], utilizing the so-called logistic map (first-order equation) with chaotic behaviors.
- **Strong-blenders** [25, 92], which provide theoretically proved guarantees for the statistical quality and unpredictability of the output.

2.3 Secure Storage for the Cryptographic Keys

Once generated, keys should be stored in secure manner, to be protected against attacker. According to sensitivity and criticality of the information, various approaches for storage keys are used today in practical applications. Naive approach is to store the key in non-volatile memory directly on the device. Though very easy to implement, this method provides no security, and should be not used for any cryptographic purposes. Another variation of this method is to store the key on the other device, but cannot be considered as more secure.

Other approach is to store key in volatile memory only. Once the device starts up, the encryption key has to be typed in by the operator. Another variation of this approach, that is often used today, is to store the keys in encrypted form in non-volatile memory, and use the user's password for encryption/decryption. Thus user logs on with his password and the password is used to decrypt the keys in the non-volatile memory. The passwords in plain decrypted form are present only in volatile memory, while in non-volatile memory they are stored encrypted. This approach protects against offline attacks, but still leaves the keys vulnerable to various on-line attacks and side-channel attacks, that are targeted on implementation of the cryptosystem itself, however, still may be suitable for common usage in non-critical applications, as efficient trade-off between security, implementation complexity and usability.

Variations on this principle were presented in past. In [9] author discuss possibilities of file-system encryption for UNIX operating system, where each directory is protected by set of cryptographic keys. The pass-phrases used for key generation are either entered by user via keyboard, or can be stored on removable media. The keys are then computed from the provided pass-phrase, that must be of sufficient length to allow the creation of several independent keys. Principle is also utilized by modern operating systems - for example, iOS encrypts content of file with a per-file key, which is wrapped with a class key and stored in file's meta-data, which is in turn encrypted with file system key. The class key is protected with the hardware unique identifier and the user's pass-code [1]. Similarly, in EFS implemented in Windows operating system for file encryption, master key for file encryption is encrypted with password encryption key, that is produced by HMAC and SHA-1 - password encryption key is hash of the user's master key produced encrypted by 160-bit RC4, user's security identifier and user's logon password [75]. In Android operating system, key store intended to store user certificates and private keys is organized as follows: master key is encrypted with a 128 bit AES key derived from the screen unlock password by applying the Password-Based Key Derivation Function 2 (PBKDF2, [45]) key derivation function with 8192 integrations and randomly generated 128-bit salt. Key blobs (binary large object) contains serialized key along with meta-data, initial vector used for encryption and an MD5 hash value of the data, all that encrypted with master key using 128 bit AES in CBC mode [27].

The most secure and thus recommended approach for critical and sensitive applications is to utilize secure hardware storage.

In [94] authors discuss that even if using provably secure cryptographic algorithms, they are dependent on secret information such as PINs, keys etc. Thus this secrets should be securely stored, preferably in some form of security hardware module.

Various tamper-resistant techniques for embedded systems were presented. In [88] authors discuss the following objectives:

1. **Attack prevention**, including techniques such as physical protection mechanism, hardware design (circuit implementation whose timing and power characteristics are data independent) and software design (software authentication before execution).

2. **Attack detection**, e.g. run-time detection of illegal memory access,

3. **Attack recovery**, e.g. locking up the system and rendering it useless, zeroing out sensitive data in the memory, or rebooting the system,

4. **Tamper evidence** to preserve an irrefutable, persistent record of an attack.

Based on this requirements, Hardware Security Modules (HSM) are nowadays used for most critical applications. HSM is, as discussed in [109], physically secure, tamper resistant security server, that provides cryptographic functions, such as random number generation, key generation, asymmetric private key storage (while providing protection from various attacks, e.g. no unencrypted private keys in software or memory), encryption/decryption and signing. All the cryptographic operations take place in HSM, so as the cryptographic keys never leave it. As further discussed in [73] to provide high-grade cryptographic security following measures should be applied:

1. Tamper-detection mechanism, which must be active regardless of the HSM's power state,
2. Physical barriers to make successful tampering infeasible,
3. Sufficient resistance to tampering, so that successful tampering requires an extended period of time,
4. The HSM's construction is such that successful tampering will cause visible damage to the device.

However, to implement such measures is complex and cost-ineffective task, therefore often neglected in practical applications in small IoT devices and embedded applications.

2.4 Physical Unclonable Functions

Though TRNG are mostly used nowadays for cryptographic key generation, in recent years, numerous works dealing with PUFs for key generation had been published, proposing PUFs as another possible approach for key generation. Moreover, PUFs usage is promising to solve the issue of secure storage of cryptographic keys. Instead of storing the key in memory, the key is generated at the time it is needed.

This radically new approach to secure key storage utilizing PUF is defined in [38]. With regards to drawbacks of non-volatile storage, authors define the criteria for new approach:

1. Key should not be permanently stored in digital form on the device,
2. Key should be extracted from the device only when required, and after having been used, it should be removed from all internal registers, memories, and locations,
3. Key should be somehow uniquely linked to a given device such that it cannot be reproduced or the device with a same key manufactured.

Thus, the keys should be extracted from the intrinsic properties of the device, and the authors further build their concept for secure key storage on Physical Unclonable functions rather than storing keys in non-volatile memory.

The concept of PUF was originally introduced in [84] and the authors showed that instead of relying on number theory, the mesoscopic physics of coherent transport through a disordered medium can be used to allocate and authenticate unique identifiers by physically reducing the medium's micro-structure to a fixed-length string of binary digits. These physical one-way functions are inexpensive to fabricate, prohibitively difficult to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. This makes PUF as ideal candidate for providing tamper resistant design for cryptographic key generation and storage. According to [63] PUF is best described as an expression of an inherent and unclonable instance-specific feature of a physical object and as such has a strong resemblance to biometric features of human beings, line fingerprints, and thus cloning a PUF is extremely hard if not impossible at all.

PUF is a system responding to a challenge C with a response R, referenced in general as a challenge-response pair. In [36] following assumptions are summarized:

- It is assumed that a response R_i to a challenge C_i gives only a negligible amount of information on another response R_j to a different challenge C_j with $i \neq j$.
- Without having the corresponding PUF at hand, it is impossible to come up with the response R_i corresponding to a challenge C_i, except with negligible probability.
- Finally, it is assumed that PUFs are tamper evident. This implies that when an attacker tries to investigate the PUF to obtain detailed information of its structure, the PUF is destroyed. In other words, the PUF's challenge-response behavior is changed substantially.

Based on challenge-response behavior, the strong and weak PUF concept was introduced in [36]. Strong PUF has so many challenge-response pairs available for PUF such that an attack based on measuring the challenge-response pairs has only insignificant probability to succeed ($1/N \approx 2^{-k}$ for $k \approx 100$), where N is the number of challenge-response pairs. Therefore, it is not possible to build a model of the PUF by predicting additional challenge-response pairs. If the number of different challenge-response pairs is small we consider it weak PUF.

2.4.1 PUF Evaluation

There is not much publications dealing with thorough and formalized approach to the PUF testing, however, there are few metrics that are commonly being measured and evaluated when new PUF design is being proposed.

A PUF response intra-distance is a metric describing the distance between two PUF responses from the same PUF instance to the same challenge:

$$D^{intra}_{puf_i}(x) = dist[Y_i(x); Y_{i+n}(x)], \tag{2.5}$$

with $Y_i(x)$ and $Y_{i+n}(x)$ two distinct and random PUF responses with the same challenge x and $dist$ being most often a Hamming distance of the two responses, represented as a bit vectors. The intra-distance is used to evaluate the similarity of PUF outputs, or the reproducibility of challenge-response pairs.

A PUF response inter-distance is a metric describing the distance between two PUF responses from different PUF instances to the same challenge:

$$D_P^{inter}(x) = dist[Y_i(x); Z_i(x)], \tag{2.6}$$

with $Y_i(x)$ and $Z_i(x)$ two responses of the two distinct PUF instances with the same challenge x and *dist* being most often a Hamming distance of the two responses, represented as a bit vectors. The inter-distance is used to evaluate uniqueness.

Further, the identifiability of a PUF is defined as probability that PUF responses to same challenge from single PUF instance are more similar compared to the PUF responses from multiple different instances.

The PUF responses for multiple challenges should be unpredictable, and thus their randomness should be also evaluated. Similar approach as for the testing of TRNGs can be adopted for this task.

The formal approach on how to test PUF designs was introduced in [69]. Authors identified the possible attacks on the PUFs and proposed the testing methods that are dictated by them:

- **Prediction attack** - An adversary who has access to the PUF or to a partial database of challenge-response pairs (CRPs) may attempt to predict the response to a new challenge by studying the probability, conditional probability, or Hamming distances among the observed CRPs,
- **Reverse engineering attack** - The attacker may attempt to model the input/output behavior of the system by studying a set of CRPs, the attack may include also the utilizing of knowledge of the PUF architecture,
- **Collision attack** - Collision occurs if two different challenge sets C_1 and C_2 produce the same response on the PUF.

Based on the possible attacks, the proposed testing methodology covers four tests:

- **Predictability test** - looking for the CRP relationships such that the response to a new challenge can be guessed with higher than random probability. This includes testing single bit probability (targeting each individual response bit, the expected probability is 1/2), conditional probabilities (conditional probability of the response bits with respect to the other response bits and with respect to the input) and Hamming distances (practically the intra- and inter-distances).
- **Collision test** - determining the probability of how often a pair of PUFs will generate same responses to given challenges. The PUF responses should be uniformly distributed.
- **Sensitivity test** - targeting the sensitivity of PUF responses to component imperfections, defects, variations of operational conditions, and ambient noise.

- **Reverse engineering test** - complex interactions among the finite number of components is what defies the reverse engineering attack and linear system is breakable by using linear optimization, e.g. using a linear number of challenge-response pairs and forming a system of linear inequalities may be used.

Another evaluation methodology was presented in [49], here authors focused on two major areas:

- **Robustness analysis** - the robustness is quantified by Bit Error Rate

$$BER = \frac{dist[Y_i, Y_{enroll}]}{|Y_{enroll}|}, \tag{2.7}$$

  which indicates the number of bits of a response Y_i that are different from the response Y_{enroll} observed during enrollment,

- **Unpredictability analysis** - the unpredictability ensures that response for particular challenge can only be guessed or calculated with negligible probability. For this purpose, the Shannon entropy (equation 2.4) is used, authors check whether PUF responses are biased by computing their Hamming weight and estimate an upper bound of the entropy of PUF responses using a compression test.

2.4.2 Physical Unclonable Functions Designs

Variety of PUF designs were proposed, utilizing random processes and variations introduced during manufacturing process, including the ones based on completely non-electronic features of materials like the optical PUF proposed in [84] or coating PUF proposed in [111]. However, these are not always suitable for use in simple devices and reconfigurable microcontrollers. In [36] the concept of intrinsic PUF was introduced. Intrinsic PUF is defined as PUF generating circuit already present in the device and that requires no modification to satisfy the security goals. As further elaborated in [62], this essentially means the evaluation of the PUF is performed internally by embedded measurement equipment and its random instance-specific features are implicitly introduced during its production process.

The [62] describes three classes of intrinsic PUFs based on their operating principles.

- **Delay-based silicon PUFs** based on measurement of random variations on the delay of a digital circuit,

- **Memory-based silicon PUFs** utilizing random parameter variations between matched silicon devices, also called device mismatch, in bistable memory elements,

- **Mixed-signal circuits PUFs**, consisting of mixed-signal circuits with basic operation is of an analog nature, which means an embedded analog measurement is quantized with an analog-to-digital conversion to produce a digital response representation.

The first big group of delay-based PUFs are arbiter PUFs, first introduced in [34] and based on a race condition between two digital paths that act as the two inputs into the arbiter circuit, both low initially. The arbiter waits for one of the inputs to go high, at which time its output indicates which input went high first. In the figure Fig. 2.13, the structure and operation of the circuit is shown - two signals race through the delay paths that are defined by the challenge input bits. At the end of the circuit, the arbiter decides which signal arrived first and outputs a bit. The arbiter circuit behaves like the MAX circuit except that the AND gate is replaced by an arbiter. However, this approach is not suitable for all technologies, on FPGA symmetry requirements cannot be satisfied using available FPGA routing schemes, despite the apparent routing flexibility of FPGA devices as shown in [78], however, as directly proposed, an architecture without the mirror symmetry requirement, such a RO based PUF (ROPUF) is more suitable for FPGA devices.

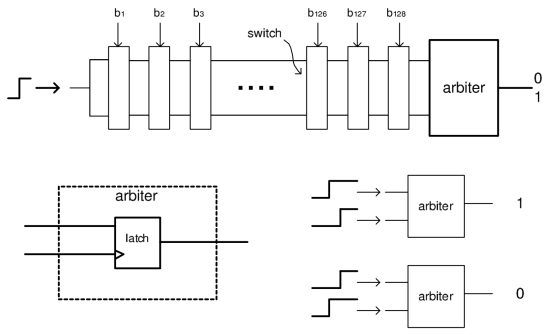


Figure 2.13: The arbiter circuit as introduced in [34].

The ROPUF design was first proposed in [33], followed by number of different designs. The original design [33], depicted in Fig. 2.14, is based on the variant of the switch block-based delay circuit, similar one as for the arbiter PUF discussed above, with added feedback. The idea of ROPUF was further utilized in additional designs. TERO design [13] builds on TERO cells, originally proposed for building TRNG. PUF delay circuit [106] consisting of identically laid-out delay loops (ROs) utilizes the fact that due to manufacturing variation each of RO oscillates with a slightly different frequency. In order to generate a fixed number of bits, a fixed sequence of oscillator pairs is selected, and their frequencies are compared to generate an output bit. Since there may appear negative effects caused by environmental noise and systematic variations in the die, the methods how to address these negative effects for ROPUF designs were discussed in [66, 67]:

- Place the group of ROs as close as possible to each other e.g. in a 2D array formation on the FPGA.
- While selecting the RO pair to read out the responses, pick the pair of ROs such that they are located adjacent.

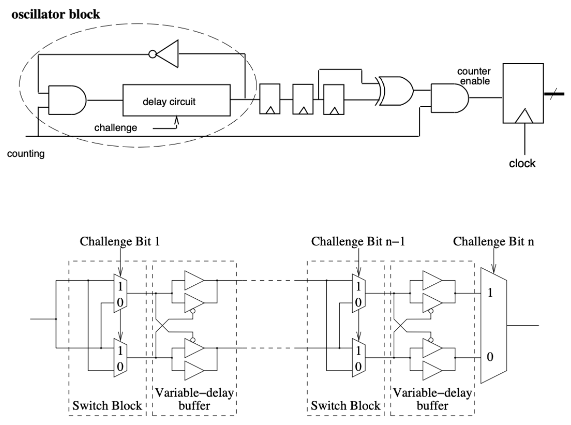


Figure 2.14: The oscillator block and non-monotonic delay circuit from ROPUF [33] design.

Memory-based SRAM PUFs were originally introduced in [36] to resolve the problem of IP protection on reconfigurable hardware. The design depicted in Fig. 2.15 utilizes the CMOS SRAM cell consisting of a six transistors. formed of two cross-coupled inverters (load transistors PL, PR, NL and NR) and two access transistors (AXL and AXR) connecting to the data bit-lines (BLC and BL) based on the word-line signal (WL). The transistors forming the cross-coupled inverters (PR,PL, NR and NL) are constructed particularly weak to allow driving them easily to 0 or 1 during a write process. Hence, these transistors are extremely vulnerable to atomic level intrinsic fluctuations which are outside the control of the manufacturing process and independent of the transistor location on the chip. During power-up, the cross-coupled inverters of a SRAM cell are not subject to any externally exerted signal. Therefore, any minor voltage difference that shows up on the transistors due to intrinsic parameter variations will tend toward a 0 or a 1 caused by the amplifying effect of each inverter acting on the output of the other inverter. Hence with high probability an SRAM cell will start in the same state upon power-up. Very similar idea was also presented in [41].

SRAM arrays on most commercial FPGAs (when present) are forcibly cleared immediately after power-up. This results in the loss of any PUF behavior, thus a design mimicking the SRAM PUF behavior the Butterfly PUF was introduced in [59], depicted in Fig. 2.16. The behavior of an SRAM cell is mimicked in the FPGA reconfigurable logic by cross-coupling two transparent data latches, forming a bistable circuit. However, similarly as in case of delay-based arbiter PUFs, the butterfly design was evaluated in [78] with the same result that the required symmetry cannot be achieved in FPGA devices, and as such the design is not suitable for FPGA technology.

Apart from SRAM, any bi-stable memory element can be utilized for a PUF design, examples include Latch PUF [105], Flip-Flop PUF [64] or Buskeeper PUF [104].

Mixed-signal PUFs utilize primarily the analogue source for generating PUF response,

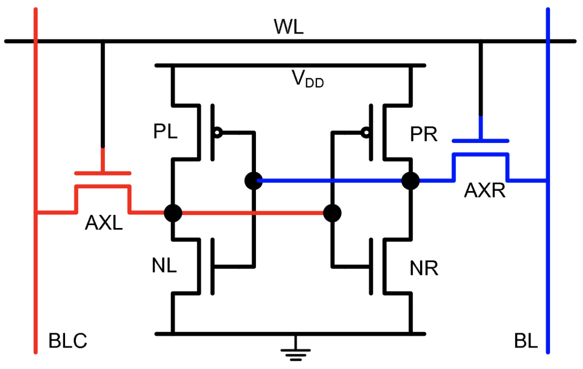


Figure 2.15: Six transistor SRAM cell [36] as used for building CMOS SRAM PUF.

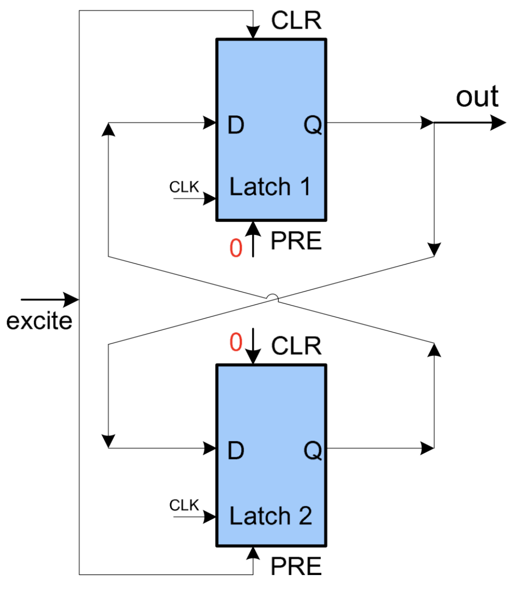


Figure 2.16: Cross-coupled latches in Butterfly PUF [59].

and therefore require analog measurement components and analog to digital converters. In [60] Integrated Circuit IDentification (ICID) block was proposed for IC identification circuit device (depicted in Fig. 2.17, based on an array of addressable MOSFETs, with common gate and source and sequentially selected drains, driving a resistive load. Because of device mismatch, the drain currents will be randomly different, producing a sequence of random voltages across the load. ICID uses these sequences of random but repeatable voltages to construct unique identifications. The sequences are different for every die, because every transistor has a different distribution of frozen-in random dopant atoms. The design was proposed even before the PUF concept was introduced by Pappu.

Another example of mixed-signal PUF is also utilizing MOSFETs [87], based on two-dimensional array of inverters. Each bit of the response is obtained as a function of the analog output of a number of inverters biased around the point of maximum gain. The digital challenge is a sequence of N inverter addresses, the response bits are obtained from a group of consecutive addresses, through a bit extractor circuit. Direct measurement of MOSFET characteristics and features allowed number of other designs to be developed, such as [101], [46], [76] and others.

Another approach to the mixed-signal PUFs include low power current-based PUF [68], converting the analog variations present in device leakage currents to a unique digital quantity, memristor-based PUFs [91] or dedicated analog designs to be used as PUF - skew memories and massively parallel analog processor arrays (CNNs)[17].

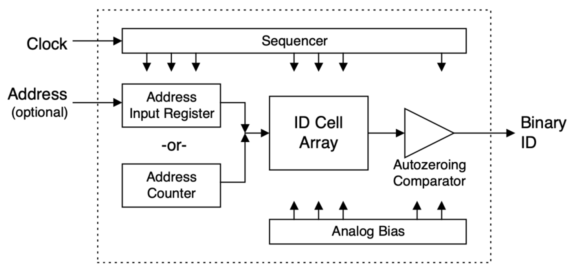


Figure 2.17: Block Diagram of ICID block [60].

2.4.3 PUF-Based Implementations and Protocols for Authentication and Secure Communication

To fully utilize the PUFs as a primitives for building the secure cryptosystem, the design and implementation has to consider all the characteristics, advantages and disadvantages of PUFs. Mere PUF design is simply not enough, the wrapping algorithms and protocols has to be developed around it. The PUF responses in real world implementations are noisy, and not always ideally stable, therefore, the post-processing is key part of the design to allow the reliable generation of cryptographic keys.

In the [65] the PUFKY - PUF-based cryptographic key generator design was proposed. It uses a highly optimized ROPUF design, based on [125], overall PUFKY design depicted in Fig. 2.18. Response generation is based on the frequency ordering of a set of oscillators, as depicted in Fig. 2.19. To amortize the overhead of the frequency counters, oscillators are ordered in b batches of a oscillators sharing a counter. In total, the PUFKY ROPUF design contains $b \times a$ oscillators of which sets of b can be measured in parallel.

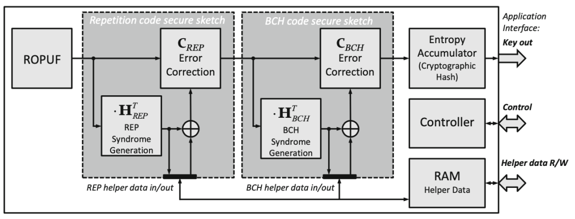

ROPUF
Repetition code secure sketch
C_{REP} Error Correction
$\cdot H^T_{REP}$ REP Syndrome Generation
BCH code secure sketch
C_{BCH} Error Correction
$\cdot H^T_{BCH}$ BCH Syndrome Generation
Entropy Accumulator (Cryptographic Hash)
Controller
RAM Helper Data
Application Interface:
Key out
Control
Helper data R/W
REP helper data in/out
BCH helper data in/out

Figure 2.18: PUFKY: PUF-based cryptographic key generator architecture [65].

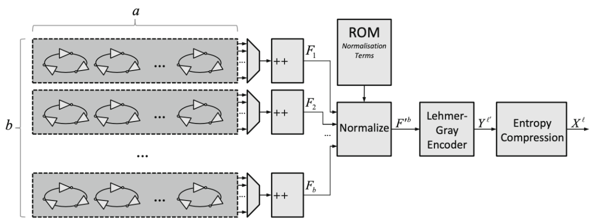


Figure 2.19: PUF-based cryptographic key generator(PUFKY): ROPUF architecture [65].

Efficient, yet still secure post-processing of generated responses was the main challenge for the authors of PUFKY, as the methods for generation of reliable keys, like fuzzy extractors introduced in [26] and further used in [12], [113], [74], [47] and many others may not be always preserve the entropy of the generated responses. The problematics of post-processing the PUF responses was studied in[58], usually consisting of conventional error correcting code such as repetition code, BCH, Reed-Muller, or Golay code. In [58] it is showed that fuzzy extractors based on repetition codes are not without risk, particularly when the PUF response has low entropy.

Therefore, the authors of the PUFKY propose the three step post-process:

1. Frequency normalization to remove structural bias from the measurements,
2. Order Encoding to encode the normalized frequency ordering to a stable bit vector in such a way that all ordering entropy is preserved, using Gray and Lehmer encoding,
3. Entropy Compression to oppress the order encoding to maximize the entropy density without significantly increasing the bit error probability, using simple XOR corrector.

The first attempts for using the PUF in the authentication schemes, were rather simple and naive. In [106] simple authentication against authentication authority was discussed, using pre-generated challenge-response pairs stored centrally. At the authentication time, the challenge is sent to the device and response then compared with the output, as depicted in Fig. 2.20. Same challenge cannot be reused again due to possible replay attacks, thus limiting the number of possible authentication attempts. Moreover, the simple protocol only allows authentication to central authority, it does not allow mutual device authentication.

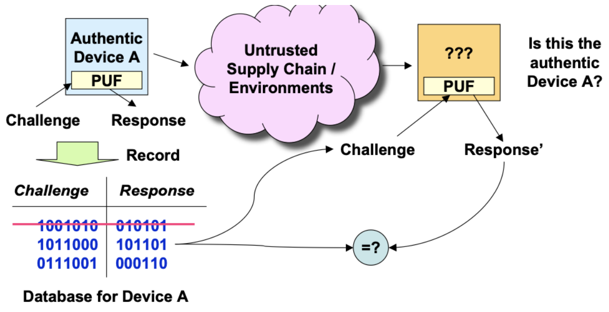


Figure 2.20: PUF-based authentication as per [106].

The lightweight challenge response authentication scheme based on PUFs was presented in [82], suitable for low cost devices such as RFIDs, sensor network nodes, and smart cards, consisting of PUF enrollment phase (authentication server collects challenge-response pairs from the circuit) and authentication phase (authentication server randomly queries responses for selected challenges and evaluates number of errors in response). Additionally, challenges are xored with the secret vector X, generated by the additional PUF circuit to prevent the attacker to have access to challenge-response pairs being transmitted over the unprotected channel, X being incremented after each use. The protocol is listed in listings Algorithm 2.1 and Algorithm 2.2.

Algorithm 2.1 Ozturk et al. [82] authentication protocol - Enrollment Phase

```
1: Authentication Server S initializes secret vector $X^0$ inside PUF instance P to a random
   value
2: S collects $(X^j, R)$ pairs from P and uses the values to model the system
3: S disables the $X^j$ reading logic
```

Algorithm 2.2 Ozturk et al. [82] authentication protocol - Authentication Phase

```
1: S picks a random challenge C
2: P receives the challenge C, iterates $X^j$
3: P calculates $C' = C \oplus X^{j+1}$
4: P enters $C'$ into the PUF circuit
5: P sends response $R = PUF(C')$ to S
6: S accepts P if the number of errors in R is not more than expected from the noisy PUF
   circuit
```

Another authentication protocol was presented in [37], similarly as in previous cases consisting on the enrollment and authentication phases. Again, two PUFs are used and a random string is required in the authentication process as an 2-level authentication circuit. Once device enrolled, the authentication phase follows, as listed in Algorithm 2.3.

Algorithm 2.3 Hammouri et al. [37] authentication protocol - Authentication Phase

```
1: PUF instance P randomly generates the n-bit string B, and sends it to authentication server D
2: D picks a random n-bit string C and sends it to P
3: P receives the challenge C, and then calculates $R = F_{Y_0,Y_1,Y_{out},S_0,L,\epsilon}(B,C)$, where $Y_0, Y_1, Y_{out} \in R^{n+1}$, noise parameter $\epsilon \in (0,1)$, the initial bit string $S_0 \in \{0,1\}^n$ and L being LFSR function.
4: P sends the bit R to D
5: Steps 1 through 4 are repeated for k iterations
6: D accepts P iff the number of errors in the responses R is not more than $\epsilon_f k$.
```

In contrast to classical Physically Unclonable Functions, Logically Reconfigurable Physical Unclonable Functions (LR-PUFs) introduced in [50] can be dynamically 'reconfigured' after deployment such that their challenge/response behavior changes in a random manner to allow PUF instance recyclation, the concept is depicted in 2.21. Similarly as in previous cases, the enrollment phase is required. Two variants of algorithm are possible: speed optimized Algorithm 2.4 and area optimized Algorithm 2.5.

Algorithm 2.4 Katzenbeisser et al. [50] authentication protocol - Speed Optimized. $mapin_S(c)$ is an input transformation, $mapout_S(y)$ is an output transformation, $rcnf()$ function refconfigures the PUF by changing the current state S to a new independent state.

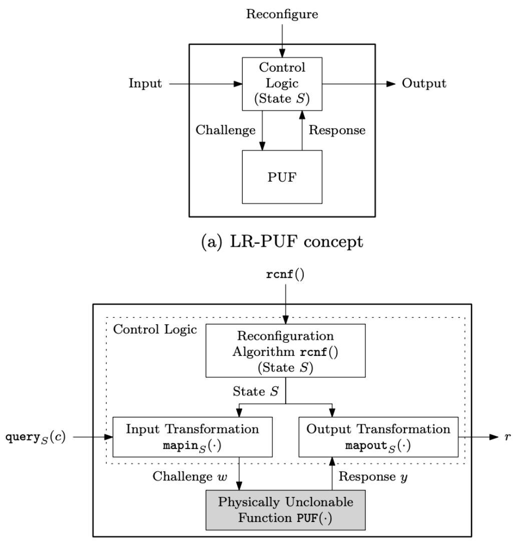
```
1: PUF is challanged with $query_S(c)$, where $c$ is a random challenge and $S$ is a state maintaned by control logic.
2:   Control logic computes $w \leftarrow Hash(S||c)$  $//mapin_S(c)$
3:   $y \leftarrow PUF(w)$
4:   $r \leftarrow y$  $//mapout_S(y)$
5: return r
6: $reconfigure()$
7:   $S \leftarrow Hash(S)$
```

Slender PUF protocol [70] does not expose the full PUF responses, only the random subset is revealed and send for authentication instead. Both prover and verifier jointly participate in producing the challenges. The joint challenge generation provides effective protection against a number of attacks. Unlike original PUF authentication methods, an adversary cannot build a database of CRPs and use an entry in the database for authentication, as described in Algorithm 2.6.

The protocol introduced in [113] differs from the protocols above - it does not deal with the authentication against central authenticating authority, but rather with mutual

Algorithm 2.5 Katzenbeisser et al. [50] authentication protocol - Area Optimized. $mapin_S(c)$ is an input transformation, $mapout_S(y)$ is an output transformation, $rcnf()$ function refconfigures the PUF by changing the current state S to a new independent state. $PUF(w_j)$ in this case is one single bit PUF that is evaluated sequentially n times to generate an n bit PUF response.

```
1: PUF is challanged with $query_S(c)$, where $c$ is a random challenge and $S$ is a state
   maintaned by control logic.
2: for $j = 0$ to $n$ do
3:     $w_j \leftarrow Hash(S||c||j)$  $//mapin_S(c)$
4:     $y_j \leftarrow PUF(w_j)$
5: end for
6: $r \leftarrow (y_0||...||y_n)$
7: return r
8: $reconfigure()$
9:   $S \leftarrow Hash(S)$
```


Reconfigure
Input
Control Logic (State S)
Output
Challenge
Response
PUF
rcnf()
Control Logic
Reconfiguration Algorithm rcnf() (State S)
State S
query_S(c)
Input Transformation mapin_S(·)
Output Transformation mapout_S(·)
r
Challenge w
Response y
Physically Unclonable Function PUF(·)

(a) LR-PUF concept

(b) Generic LR-PUF construction

Figure 2.21: Logically Reconfigurable PUFs: Concept and generic construction [50] .

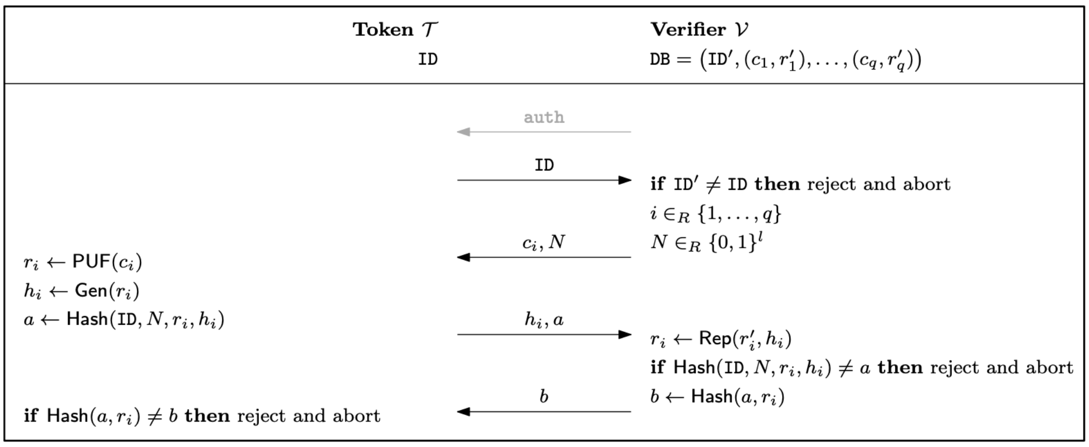
authentication of both device and authentication server. The authentication protocol as depicted in Fig. 2.22 works as follows. Verifier ν sends an authentication request $auth$ to device τ. Device τ responds with its identifier ID. ν selects a random challeng c_i and nonce N for the authentication session. ν generates an response r_i for c_i. Further, it generates value h_i using the reverse fuzzy extractor and computes $Hash$ of ID, N, r_i and h_i. ν performs the same calucation based on challenge-response pairs stored in the database for

Algorithm 2.6 Slender PUF lightweight protocol [70].

```
1: Verifier: $Nonce_v$ generated by the verifier using TRNG.
2: Prover: $Nonce_p$ generated by the prover using TRNG.
3: Verifier: $Seed = \{Nonce_v, Nonce_p\}$, Prover: $Seed = \{Nonce_v, Nonce_p\}$
4: Verifier: $C = G(Seed)$, Prover: $C = G(Seed)$, where $C$ is an random challgenge
   generated by the function $G()$ representing PRNG.
5: Verifier: $R' = PUF_model(C)$, Prover: $R = PUF(C)$ where $R$ is an PUF response.
   Verifier has an access to a secret model of a PUF $PUF_model()$.
6: Prover: $W = sub - seq(ind, L_sub, R)$, where $W$ is a substring of PUF response, gener-
   ated using the randomly chosen index $ind$ with a predefined length $L_sub$.
7: Verifier: $T = match(R', W, e)$, Authentication passes if $T = true$.
```

device τ. If both hashes match, the authentication suceeds. The proposed PUF uses reverse fuzzy extractors, that instead of implementing the computationally intensive reproduction phase, implement the much more efficient helper data generation phase on the PUF-enabled device and move reproduction phase to the typically more powerful verifier.

Protocol allowing clients to authenticate the server was introduced in [54], introducing a new paradigm of Converse PUF based Authentication Protocols: A prover P, who has access to a PUF, wants to authenticate himself to a verifier V who holds a database of challenge-response pairs of P's PUF. During the enrollment phase the database of challenge response pairs of P's PUF, in authentication phase a verifier choose a random challenge, queries the Prover's database, and compare the result with local PUF instance. Third, optional, phase of the protocol is computation of the shared key, by inputting the PUF response into the hash function for key generation.



Token $\mathcal{T}$
ID
Verifier $\mathcal{V}$
$\mathtt{DB} = (\mathtt{ID}', (c_1, r'_1), \ldots, (c_q, r'_q))$
auth
ID
if $\mathtt{ID}' \neq \mathtt{ID}$ **then** reject and abort
$i \in_R \{1, \ldots, q\}$
$N \in_R \{0,1\}^l$
c_i, N
$r_i \leftarrow \mathsf{PUF}(c_i)$
$h_i \leftarrow \mathsf{Gen}(r_i)$
$a \leftarrow \mathsf{Hash}(\mathtt{ID}, N, r_i, h_i)$
h_i, a
$r_i \leftarrow \mathsf{Rep}(r'_i, h_i)$
if $\mathsf{Hash}(\mathtt{ID}, N, r_i, h_i) \neq a$ **then** reject and abort
$b \leftarrow \mathsf{Hash}(a, r_i)$
b
if $\mathsf{Hash}(a, r_i) \neq b$ **then** reject and abort

Figure 2.22: Lightweight PUF-based mutual authentication protocol [113].

In recent research privacy begins to be concern, and as such, several privacy preserving algorithms were introduced. In [77] the definition of indistinguishability-based privacy was used - if adversary selects two devices he will not be able to distinguish the communication derived from one of the two devices. Apart from PUF output itself, the shared secret key between the server and the device, initialized during the enrollment phase, is used, auto-updating after each transaction. Similar idea of shared secret key in addition to PUF was used in [3].

Some of the PUF based authentication protocols were reviewed in [20], identifying several potential issues being commonly present across various designs:

1. **Reproducibility of the PUF responses**. Since the PUF responses are not perfectly reproducible, the protocols has to either use the error correction, or tolerate the errors, however the stability of the responses is often over-estimated.

2. **Output size space**. Strong PUFs have a small output space, that may be exploited by machine learning attacks.

3. **Modeling attacks**. PUF designs are usually easy to model, especially in cases when the response bits are non-uniformly distributed.

4. **Need of Secure TRNG**. Apart from PUF, the protocols often employ random numbers or vectors in the process of authentication.

Chapter 3

Overview of Our Approach

As presented in previous chapter, both TRNGs and PUFs have different characteristics that are advantageous in different applications. In this chapter we will discuss the requirements laid on cryptographic keys in various cryptosystems, and thus suitability of TRNG and PUF for this tasks and their efficient design. Further, the suitable protocols for lightweight authentication and secure communication for IoT and embedded devices will be discussed, take advantage from efficient and secure hardware implementations of TRNG and PUF.

3.1 TRNG Design

Different cryptographic protocols and algorithms rely on random numbers. The applications include generating session and ephemeral keys, paddings, nonces and salts etc. Therefore, the quality of random number generator has a significant influence on security of whole cryptosystem.

Strict requirements are laid on the quality of generated cryptographic key for sensitive applications, as well as meeting various mathematical properties of a particular cryptosystem. The key should be unpredictable, and thus is today most often generated using TRNG. Moreover, TRNGs are of great advantage when generating other short-term or public values. TRNGs are suitable for following tasks:

- Key generation for symmetric cryptography - session keys, ephemeral keys,
- Publicly transmitted or open values such as nonces, challenges, salts, initialization vectors,
- Generation of ephemeral keys and other one-time values for asymmetric schemes.

Apart from TRNGs that are traditionally used for generating cryptographic keys, PUFs are being presented as a novel approach increasing physical security. As discussed in [106], safely managing and storing secrets (including cryptographic keys) in memory is difficult and expensive. Using PUF, the keys are not physically stored, rather they are generated when they are needed. The fact that the key was generated directly on the device and cannot be copied nor cloned, is advantageous especially in the following cases:

- Identification and device authentication,
- Key generation and key storage for symmetric cryptography (for on-chip encryption)
- Key generation and key storage for asymmetric cryptography (private keys).

As such, it is advantageous for proper implementation of various cryptographic protocols to have the ability to generate both random number (e.g. for symmetric keys), as well as PUF (for secure generation and storage of asymmetric keys) at the same time.

In [A.3] and [A.5] the TRNG design based on ROPUF was proposed ([56], [57]) to prove that it is possible to create a universal circuit for generating PUF and TRNG at the same time, for further cryptographic applications. Both PUF and TRNG utilizes ROs as an entropy source. The idea to use ROs as the source of an entropy is not new. The principle of TRNG generation based on ROs is discussed in [112] - The delay instability of logic gates (inverters) connected into the ring is seen as a frequency/phase instability (a jitter) of the generated clock. The jitter is extracted in a sampling unit using flip-flops or latches. The principle is depicted in Fig. 3.1 - the basic building element of the proposed ROPUF/TRNG design is a five stage RO (1 NAND, 4 inverters). Instead of measuring frequency of each RO using reference clock one RO pair is chosen and count their oscillations simultaneously using two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing. Obtained measured counter values are used as output and they are not modified in any way. The work in [56], [57] shows that these values are represented in binary code, the appropriately selected part of each binary number for PUF output. Further, the [56], [57] assumes that if we make multiple measurements of one RO pair, bits that are close to the Least Significant Bit (LSB) will vary a lot for example due to environmental changes. On the contrary, bits close to the Most Significant Bit (MSB) will be stable and the environmental changes will have almost no influence on them.

The more we will be close to the MSB, the more stable these bits will be. Therefore, the least significant bits are suitable for random number generation. This was evaluated and confirmed using NIST test suite [6]. It showed up that 3 bits from each run of the circuit have satisfactory quality, except for uniform distribution of 1's and 0's. However, this often happens, when dealing with TRNG. There are various ways of post processing to deal with this problem, if we assume that the bias is the only problem, and the generated bits are statistically independent [21]. To enhance the properties of generated bit stream, Von Neumann and XOR post-processing methods were used – both showing satisfactory results. Final tests after applying post-processing shown that up to 142×3 random bits can be gained in one run of the ROPUF, but further post processing is needed, which shortens the generated bit stream by 50% (XOR corrector), or by approx. 75% (von Neumann corrector). Final results of NIST tests after applying correction are summarized in Tab. 3.1.

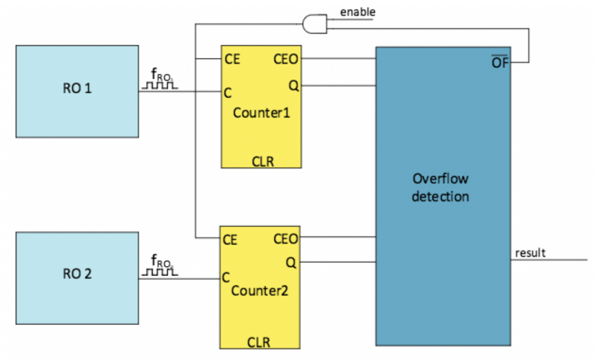


Figure 3.1: ROPUF TRNG - principle of operation.

The properties of this design were further discussed in [A.4] – with regards to temperature dependence. Influence of temperature on the proposed ROPUF for two different approaches, asymmetric and symmetric ROs. Before the experiments were carried out, i.e. for stable environmental conditions, we compared statistical properties of both approaches and the results indicated almost no differences. However, it was shown that the stability of the PUF outputs are almost equivalent for small changes in temperature, however, when the change of temperature is greater, the ROPUF using symmetric ROs gives better results.

3.2 TRNG Testing

Since the cryptographic protocols rely on the randomness and various attacks misuse not properly implemented RNGs, it is crucial to have confidence into the generated sequence. In [A.1] the methodology for testing the true random number generator implemented on an Atmel AVR micro-controller, ATmega169P, was proposed. The design based on jitter of the RC oscillator was proposed in [40], but the generated sequences were not tested thoroughly and the original tests suffered from lack of data. The principle of examined design is depicted in. Fig. 3.2 The tests included some customized test with regards to design of TRNG, as well as standardized battery of tests.

First of all, raw unmodified data were examined. As there was a possibility of component frequencies in the generated bit stream, the discrete Fourier transform was applied to confirm, that there are no such component frequencies influencing the generated sequence. Another test for detection of possible periodically repeating peaks was based on simple idea. Every $n - th$ number (for n from 2 to 512) was picked up, and arithmetic mean of each sequence was calculated. The resulting arithmetic means were than compared to arithmetic mean of original non-modified sequence. If there are any periodically repeating peaks, then some of the arithmetic means will be higher than the mean of the whole sequence. Considering the results of Discrete Fourier Transform and the mean test, it was

Test	Von Neumann	XOR
Frequency	99/100	98/100
Block Frequency	99/100	100/100
Cumulative Sums I	100/100	99/100
Cumulative Sums II	99/100	97/100
Runs	99/100	100/100
Longest Run	99/100	99/100
Rank	98/100	99/100
FFT	99/100	98/100
Non Overlapping template	97-100/100	97-100/100
Overlapping Template	100/100	100/100
Universal	100/100	100/100
Approximate Entropy	100/100	100/100
Random Excursions	62-63/63	55/55
Random Excursions Variants	62-63/63	55/55
Serial I	99/100	100/100
Serial II	100/100	98/100
Linear Complexity	100/100	97/100

Table 3.1: Testing of TRNG based on ROPUF circuit: Final results of NIST Statistical test suite after applying Von Neumann and XOR Correction for concatenated bits 1-3.

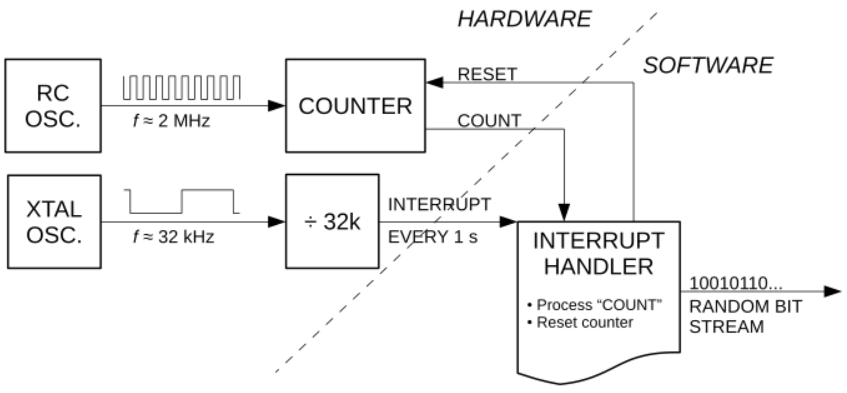


Figure 3.2: The jitter of the RC oscillator is the source of entropy in used method. The crystal oscillator was used to time a constant period (1 second), and the number of cycles of the RC oscillator that occur during that time was count.

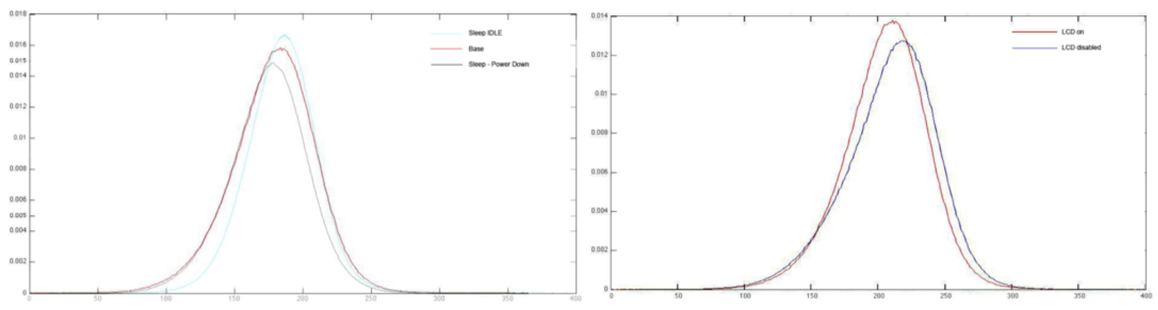


Figure 3.3: Influence of sleep modes and LCD on generated values – pictures show histograms of generated values. In the left the influence of sleep IDLE mode (light blue) and sleep - power down (black) is depicted compared to normal operation (red). In the right the operation with LCD on (red) is compared with LCD disabled (blue).

possible to say that there are no detectable component frequencies and no periodically repeating peaks.

Next, it was necessary to determine, which bits of the generated 16-bit samples have sufficient entropy and are secure to use. For those purposes, 16 separated bit streams were created, n-th bit stream consisting of n-th bit from each 16-bit sample. Then the entropy and randomness of each of the bit streams was tested, using batteries of test – Ent[120] and Diehard [72]. The tests shown, that bit 1 has sufficient entropy and suitable qualities for cryptographic use.

The design was further elaborated in [A.8] – the paper discussed the possibility of using the same method based on jitter of RC oscillator on simpler micro-controllers, as they lack dedicated TRNGs. The goal was to implement reasonable TRNG in firmware, with no or minimum added hardware. The simple module based on ATmega8 (one of simplest devices in the ATmega series) was created and generated sequence tested. We proved that method still works, however, significantly less entropy is available (generating 4 samples per second on ATmega8 provides similar entropy as when generating 128 samples per second on ATmega169P). We also investigated influence of on chip-components, sleep modes (shown in Fig. 3.3) or USART speed on generated sequence but no significant effect was observed – therefore the TRNG showed to be secure to use with various applications.

3.3 Attacks on TRNG

To demonstrate that it is important not only test during TRNG design, but also continuous self-testing of random number generators is inevitable, we presented possibilities of attack on TRNG [A.2]. Frequency injection attack presented, aimed to "stabilize" unstabler RC oscillator used as source of entropy on ATmega TRNG and thus to reduce the quality of generated bit-stream. The reasoning is that by using a relatively stable oscillator operating near the nominal operating frequency (8 MHz) of the RC oscillator, said RC oscillator

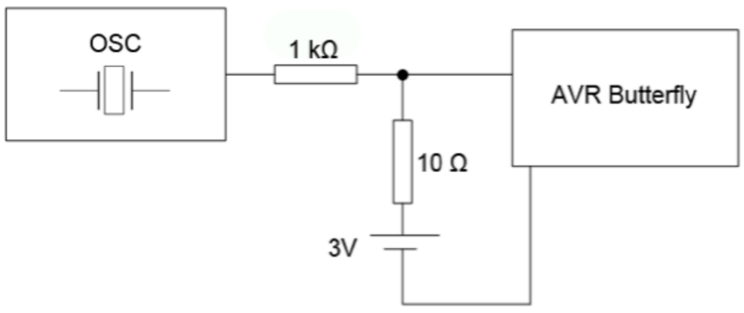


Figure 3.4: Scheme of external power supply used for invasive variant of TRNG attack

would lock to this externally applied, parasitic frequency. Consequently, it was expected there would be less variation in generated values. Two variants of attack were proposed – invasive and non-invasive.

In the invasive variant of the attack, as, a direct access to the device and its power source are needed. We attempted to inject a frequency of 8 MHz from a relatively stable source directly into the device's power supply connection. The injected frequency is generated by a simple 8 MHz crystal oscillator consisting of a crystal resonator (30 ppm stability) and inverting Schmitt triggers. The oscillator is then connected to an external power supply of the AVR Butterfly board, as depicted in Fig. 3.4. When examining the generated values, we found out that under normal operation with unmodified power supply, 6493 unique values were generated – as shown in Fig. 3.5. With the modified power supply and 8 MHz external oscillator, only 2167 unique values appeared in the generated sequence of the same length. From previous work [A.1, A.8] we knew that bit 1 of each generated value is truly random and secure for use even in sensitive applications. However, with the attack, we succeeded in influencing this bit.

Because gaining direct physical access to the device and modifying it may not always be possible, we tried a non-invasive variant of the attack, too. In this variant of the attack, a function generator was used to generate an 8 MHz sinusoid signal. A simple antenna was connected to the generator and placed near the AVR Butterfly board with the random number generator running. The non-invasive variant of the attack is not as powerful because it only affected the higher bits of the generated values, the statistical properties of bit 1 were not significantly influenced. Still, significantly less unique values were generated under the attack. One should also consider the fact that no physical manipulation with the device was necessary and no physical traces of the attack were left when it ceased.

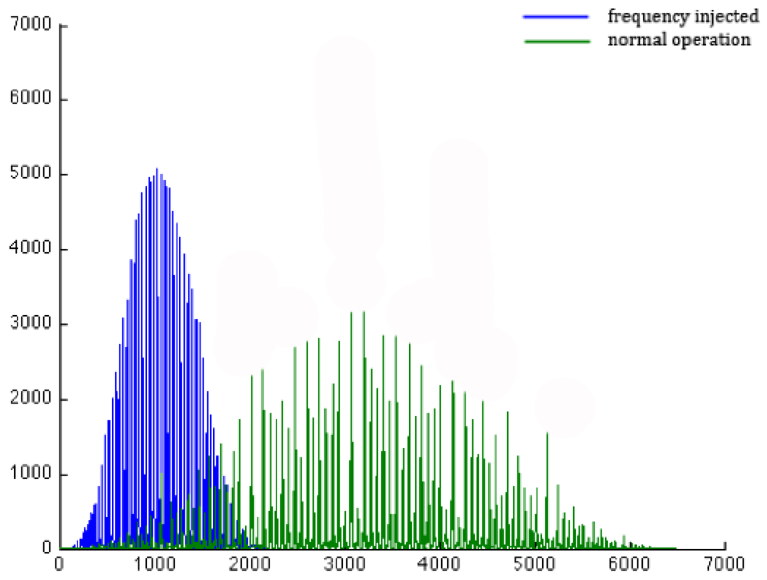


Figure 3.5: Comparison of histograms of generated values in normal operation and under the invasive variant of attack

3.4 Key Generation for Authentication and Secure Communication

Strict requirements are laid on the quality of generated cryptographic key for sensitive applications, as well as various mathematical properties of particular cryptosystem has to be met. Guidelines for key generation can be found e.g. in [5]. The key should be unpredictable, and thus is today most often generated using TRNG. Moreover, TRNG are of great advantage when generating other short-term or public values.

Symmetric cryptography utilizes the same key for both encryption and decryption and also encryption and decryption functions are very similar or even the same. Thus for exchanging encrypted messages, both sender and recipient have to share the same encryption/decryption key. For most of the scenarios involving symmetric algorithms TRNG is more suitable for key generation, as multiple different keys with high entropy can be generated. The biggest advantage of the PUF, that the key cannot be easily extracted or copied, has no meaning in this case, as the generated key has to be shared with the recipient. In the information-theoretically secure cryptosystem, that cannot be broken by adversary (Vernam cipher or One Time Pad, introduced in [118]), cipher text provides no information about plain text. To achieve this, the key has to be of same length as plaintext, and truly random. As symmetric cryptosystems are efficient and fast, they are used for encryption; asymmetric cryptography is then utilized to encrypt the symmetric – session key itself. Session keys should be periodically re-generated and are not stored in non-volatile memory in a long term, and as such the TRNG is ideal for this case. However, there is one scenario where the PUF is efficient in combination with symmetric cryptography – encryption of device's non-volatile memory, e.g. computer hard-drive encryption, as in such case the decryption key is meant to never leave the device.

Asymmetric (or public-key) cryptography is relatively new approach when compared to symmetric cryptography. While symmetric cryptosystems are used for more than 4000 years, public-key cryptography was introduced by Whitfield Diffe, Martin Hellman and Ralph Merkle in 1976 [24]. The idea is that the recipient's decryption key is kept private, while the encryption key is public, and can be used for encryption by anyone. Only the recipient is then able to decrypt the message using private key. Utilizing PUF for generating asymmetric keys brings much more benefits – as the decryption key should remain private it is ideal to utilize the PUF for key generation and storage, as the key remains private, never leaves the device and cannot be copied nor cloned to another device. For Diffie-Hellman key exchange, both TRNG and PUF can be utilized for choosing private key. However, if PUF is used, and thus the key is static for multiple sessions, the advantage of forward secrecy provided by rotation of the keys (if TRNG is used) is lost. Anyhow, the key compromise is more difficult (and almost impossible if properly implemented), as the key is not stored anywhere in non-volatile memory, and generated by PUF only when used. The same principle applies to both classic Diffie-Hellman scheme, as well as to scheme based on elliptic curves. By slightly modifying Diffie-Hellman scheme (primarily serving as key exchange algorithm), we come to El-Gamal encryption scheme [28]. The scheme consists of two steps – first is classic Diffie-Hellman for key exchange, second is message encryption itself. In this case one of communicating parties has static key, that is not changing and is being used for encrypting multiple messages, while the another party generates new ephemeral key for each encryption as reusing this ephemeral key will lead to cryptosystem breach (private key leak). Thus, the combination of PUF and TRNG is ideal for this case. RSA [89] cryptosystem seems to be ideal candidate for utilizing PUF capabilities. The chosen prime numbers p and q that multiplies to modulus, are the only values that need to be stored, the private key then can be generated on-the-fly using this values when needed. The PUF is ideal for generating and storing this p and q values. However, it is not easily achievable to ensure that PUF output will generate prime numbers as the PUF output is randomly determined by physical properties of the circuit and cannot be directly influenced. Thus additional post processing is needed, however can be efficiently implemented in hardware. The idea is simple – for example as the p and q will be used smallest (or biggest) prime number greater (smaller) than number generated by PUF., There is also need for random numbers in real world usage of RSA – for padding. Without padding the RSA encryption is deterministic, and cipher texts for short plaintext or plaintext equal to 0, 1, -1 is easily breakable. Another asymmetric cryptosystem – Paillier cryptosystem [83] is probabilistic algorithm, thus there is need for both secure storage of private parameters, as well as for random numbers during encryption and combination of PUF and TRNG is ideal for its implementation. Key generation starts (similarly as in RSA) with choosing two prime numbers, that needs to be kept private (and can be generated by PUF). Private key than can be computed on-the-fly from this values. For each encryption then random value has to be generated.

Specific subset of asymmetric cryptography are the digital signature schemes. RSA-PSS (Probabilistic Signature Scheme, [43]) is probabilistic signature scheme based on RSA, thus there is need for private key that can be generated securely by utilizing PUF. The randomization prevents dictionary attacks as well as fault-injection attacks, and thus calls for quality random numbers. Digital Signature algorithm is federal US government standard for digital signatures (DSS, [30]), proposed by NIST and as such is based on El-Gamal scheme. The private key can be reused multiple times, thus can be generated by PUF, that ensures its secure storage. However, for each signature it is critical to use new random ephemeral key, thus there is need for quality random numbers, similarly as in RSA-PSS. The same principle applies for ECDSA, that is based on elliptic curves.

As results from the previous paragraphs, a combined PUF/TRNG circuit seems to be a suitable alternative for the purpose of key generation and authentication in lightweight cryptographic applications, such as IoT devices and other embedded platforms. In the [A.6] we proposed the embedded module for secure authentication and communication with the main goal is to simplify the key management on the endpoint embedded device itself. Thus, we propose to utilize the single circuit for key generation using PUF and TRNG as a basic building block of the module so that there is no need to store secrets on the hardware device.

The overall module depicted in Fig. 3.6 provides PUF authentication and PUF/TRNG based key generation. For the authentication, the PUF is used, since it provides randomness intrinsically present in the device and utilizes the fact that the generated response is unique per device. Since there is a need for both static key, as well as ephemeral keys, combination of PUF and TRNG is used in this case – the PUF is used for generation of static (private) key that never leaves the device, thus utilizing all the advantages of PUF, while TRNG is used for generation of ephemeral, one-time keys, that are shared with other communicating parties.

3.4.1 Authentication against Central Authentication Authority

Simple and straightforward authentication protocol using pre-generated challenge-response pairs that can be easily implemented in hardware devices. This protocol does not require the PUF to have a large space of challenge-response pairs (it can be used even for one challenge-response pair). The authentication protocol consists of two phases – secure enrollment phase and authentication phase and is depicted in Fig. 3.7.

During the Enrollment phase, the Challenge/Response pair(s) (C, R) are measured from the targeted device and securely stored at the central Authenticating Authority (AA), that can be either integrated into the gateway or be represented by separate device that the gateway is querying during authentication process. A database DB_{Di} of the pairs (C, R) is created for each device D_i. Furthermore, the Public Key (PK_{AA}) of the AA is pre-set on the device, so as the authentication data can be securely transferred. For this purpose, an asymmetric scheme (El-Gamal) can be used, as proposed in the section above. We assume that PK_{AA} is protected against unauthorized changes (by the tamper-evidence

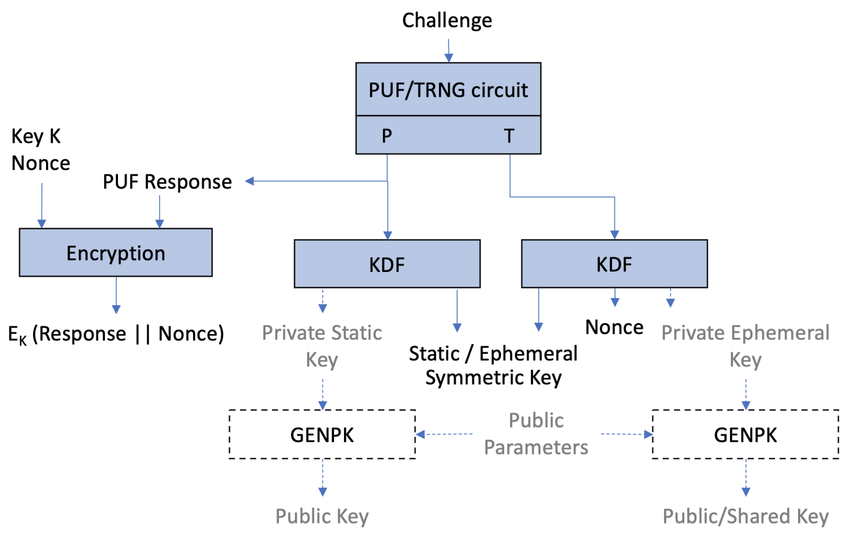


Figure 3.6: Embedded module for secure authentication and communication. KDF is a Key Derivation Function, GENPK generates a public key from a private key and public parameters. Error Correction is used to obtain a stable key material from the PUF Response.

property of the PUF). This protocol does not require the PUF to have a large space of challenge-response pairs (it can be used even for one challenge-response pair).

Every time the device is connected to the network and needs to be authenticated the authentication phase takes place. AA randomly chooses one of the challenges and sends it together with the nonce value to the device to be authenticated. The nonce value is used to prevent simple replay attacks and allows each challenge-response pair to be used repeatedly. On the device that is being authenticated the appropriate PUF response is generated, concatenated with nonce value and encrypted with public key of the AA. AA than compares the nonce value, as well as list of pre-generated challenge/response values. If both match, the device is successfully authenticated. Since the PUF response may slightly vary across various measurements, a predetermined number of faulty bits is tolerated.

3.4.2 Mutual Device Authentication

Not only the device needs to be authenticated to central authority when connected to the network, the devices must be mutually authenticated before they start to communicate, as well. Similarly, as in the previous case, central AA stores the pre-generated challenge-response pair(s), and acts as trusted 3rd party. This time though, a shared symmetric key is established between the two devices, and a conventional symmetric authenticated and encrypted session can follow afterward. The goal is to use the PUFs in both devices D_1 and D_2, but not transmit any PUF response over the network. By using the one-wayness

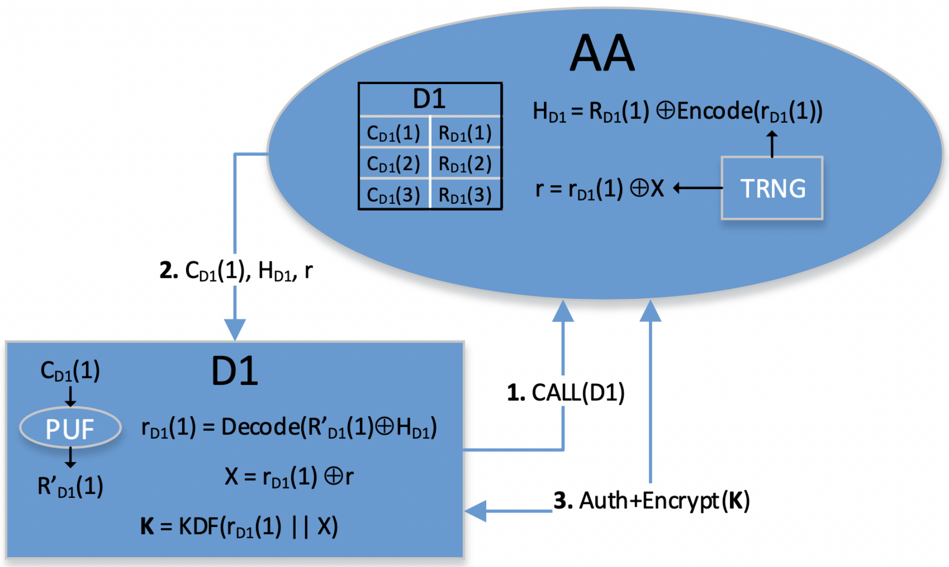


Figure 3.7: Authentication of a device D_1 to the AA.

of the hash functions used, no device gets to know other device's PUF response, even if it monitors all communication. An error correcting code is used to ensure stable PUF outputs. The codewords are selected randomly from the code space by the AA. The overall process is depicted in Fig. 3.8.

Let us assume that D_1 wants to authenticate withD_2 and set up a secure communication channel. D_1 initiates the process by calling the AA with the identification of D_1 and D_2 $(CALL(D_1, D_2))$. AA contains the complete table of challenges and responses $(C_{D1}(1), R_{D1}(1) etc.)$. An error correcting code is chosen that can correct enough errors to make the PUF response stable, with the corresponding functions Encode and Decode. AA generates two random components r_{D1}, r_{D2} from the set of preimages, and encodes them, thereby forming randomly chosen codewords. The code length should correspond to the PUF response length. Helper strings H_{D1} and H_{D2} are created by XORing the expected PUF response (R_{D1}, R_{D2}) to the corresponding codeword. The two random components are hashed and the hashes XORed to form r.

To each of the devices, a triplet (C_{Di}, H_{Di}, r) with the challenge, helper string, and r is sent. Each of the devices challenges its own PUF to get the response (R'_{D1}, R'_{D2}). By XORing the response with the corresponding helper string (H_{D1}, H_{D2}), resulting with a codeword with errors, which is then corrected by the Decode function. This way, each device recovers its component (r_{D1}, r_{D2}). D_1crecovers the value $Hash(r_{D2})$ by XORing r with the hash of its r_D1, and vice versa. This way, both devices know the hashes of r_D1 and r_D2, and can derive the shared key K by applying a key derivation function KDF on the concatenation of the hashes.

The hashing of r_{D1}, r_{D2} is done to hide the PUF responses from the other device. If D_1

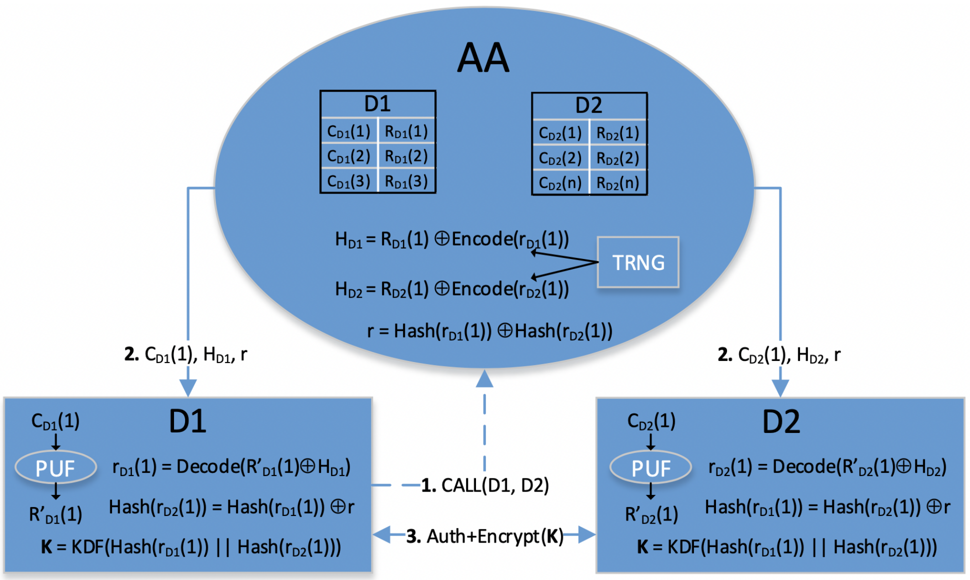

AA
D1
CD1(1) RD1(1)
CD1(2) RD1(2)
CD1(3) RD1(3)
D2
CD2(1) RD2(1)
CD2(2) RD2(2)
CD2(n) RD2(n)
HD1 = RD1(1) ⊕Encode(rD1(1))
TRNG
HD2 = RD2(1) ⊕Encode(rD2(1))
r = Hash(rD1(1)) ⊕Hash(rD2(1))
2. CD1(1), HD1, r
2. CD2(1), HD2, r
D1
CD1(1)
PUF
R'D1(1)
rD1(1) = Decode(R'D1(1)⊕HD1)
Hash(rD2(1)) = Hash(rD1(1)) ⊕r
K = KDF(Hash(rD1(1)) || Hash(rD2(1)))
1. CALL(D1, D2)
3. Auth+Encrypt(K)
D2
CD2(1)
PUF
R'D2(1)
rD2(1) = Decode(R'D2(1)⊕HD2)
Hash(rD1(1)) = Hash(rD2(1)) ⊕r
K = KDF(Hash(rD1(1)) || Hash(rD2(1)))

Figure 3.8: Mutual device authentication and secure communication.

monitors the communication, it will know $(C_D1, C_D2, H_{D1}, H_{D2}, r)$. It can recover r_{D1}, and if the hashing were not done, and r would be equal to $r_{D1} \oplus r_{D2}$ directly,D_1would compute r_D2, and using the helper string H_{D2}, it could discover the PUF response R_{D2}. We would have to either trust all devices in the network or use all challenges only once and discard them. In our case, because we do use hashing ofr_{D1}, r_{D2}, D_1only gets $Hash(r_{D2})$, and the one-wayness of the hash function prevents it from discovering R_{D2}. Thus, we can reuse the challenges for future authentications.

PUF response correction code choice depends on the number of bit flips inherent in the PUF operation. The code length and codeword distance determine the number of information bits, thus the length of r_{D1}, r_{D2}, and limit the entropy contained in r. By using the same challenge with multiple random rDi, we can extract more bits of entropy from the PUF. The entropy of the PUF response correction code choice depends on the number of bit flips inherent in the PUF operation. The code length and codeword distance determine the number of information bits, thus the length of r_{D1}, r_{D2}, and limit the entropy contained in r. By using the same challenge with multiple randomr_{Di}, we can extract more bits of entropy from the PUF. The entropy of the resulting shared key K is determined by the properties of used hash functions and KDF, and the inputs. If chosen correctly, it is as high as the entropies of r_{D1}, r_{D2}. The key K is always derived from randomly chosen codewords, and therefore for the same PUF challenges (C_{D1}, C_{D2}), a different K is obtained.

3.4.3 Secure Communication of Authenticated Devices

After the authentication process described in the previous section, a shared key is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using block ciphers such as AES. A number of lightweight block ciphers suitable for embedded systems or sensor networks has been proposed, such as PRESENT [10] with an 80-bit key. This allows generating the key in a single run of PUF circuit for most of the PUF designs and implementations, with no further stretching needed.

3.4.4 Secure Communication Using Asymmetric Encryption Scheme

An asymmetric scheme provides us with benefit that there are no shared keys and the keys do not need to be transferred over secure channel. Moreover, using PUF we do not need to store the secret key on the device, instead, the key is generated during initialization phase (e.g. on boot of the device or after longer period of inactivity), when needed. We propose to use the asymmetric El-Gamal encryption scheme [28], since there are no specific requirements on private keys (unlike an RSA - where key requires more complex processing including secure prime generation), apart from the requirement that the private key is from an appropriate range, and the private key may be easily generated from random sequence by a Key Derivation Function (KDF), such as PBKDF2 [45]. For digital signatures, either El-Gamal or DSA, or even ECDSA can be used. In subsequent text, we assume appropriate keys are generated depending on chosen algorithms. A private-public key pair of each device must be generated and the public key stored in the Authentication Authority. The whole process is described in Fig. 3.9.

After the keys are enrolled in the Authentication Authority, any device D_1 can use it to establish a secure channel by requesting a fresh and authentic public key of its peer D_2 from the AA. The public key query Q_1 is created by encrypting the identity of D_2 and a random nonce N_1 with the AA's public key. The answer A_1 contains a signed triplet with the identification of D_2, its public key PK_{D2}, and the nonce N_1. The device D_1 verifies this signature using the AA's public key PK_{AA} .

The device's own private key SK_{D1} is generated using its PUF and the previously stored challenge C_1 and helper string H_1. A new symmetric key K is generated randomly using the device's TRNG and sent encrypted with the peer's public key PK_{D2} and signed with SK_{D1}.

3.4.5 Authenticity and Integrity of the Message

For the applications where the authenticity and integrity of the transmitted message is critical, the MAC (Message Authentication Code) algorithm is used. Though similar to asymmetric digital signatures schemes (that introduce the necessity of certification or other central authority and complexity of asymmetric algorithms) in terms of providing message authentication and integrity, allows more efficient implementation using symmetric

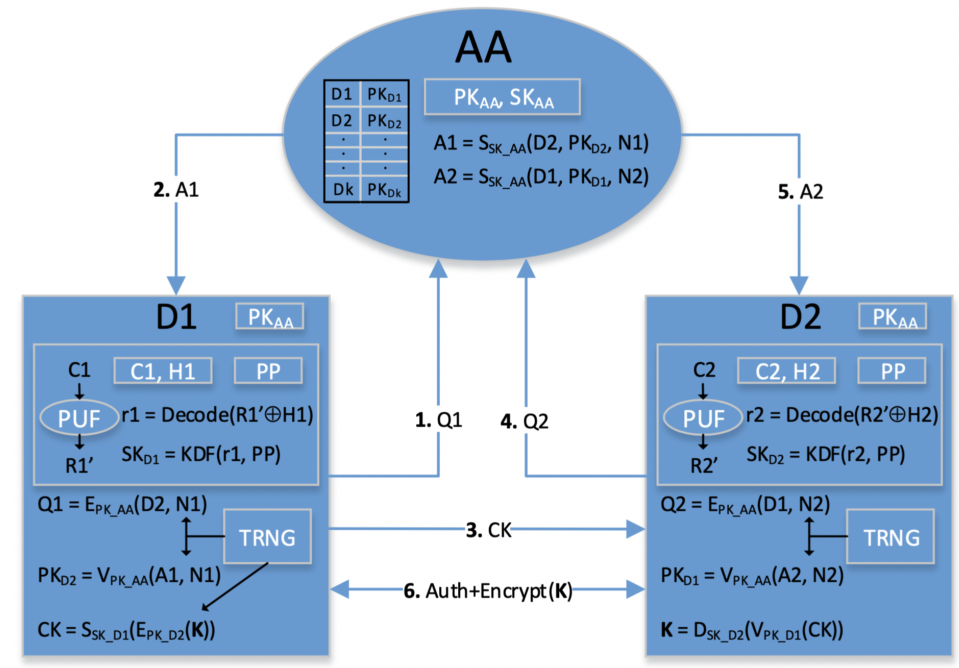


Figure 3.9: Asymmetric mutual authentication of devices. E means Encryption, D mens Decryption, S mens Signing, V means Verification..

schemes. Block cipher in cipher block chaining mode is one of the possible MAC basic building blocks, and as discussed in section above, depending on the sensitivity of the application either conventional AES algorithm or one of the lightweight schemes is chosen. Once the devices are mutually authenticated and have exchanged the shared key as discussed in the sections above, the communication may start, including message authentication.

CHAPTER 4

Main Results

This chapter presents the main results of our research in form of a collection of papers as published in the corresponding conference proceedings or journal. Fig. 4.1 illustrates the structure of the author's related papers, their relationships amongst each other, as well as their link to the thesis topic.

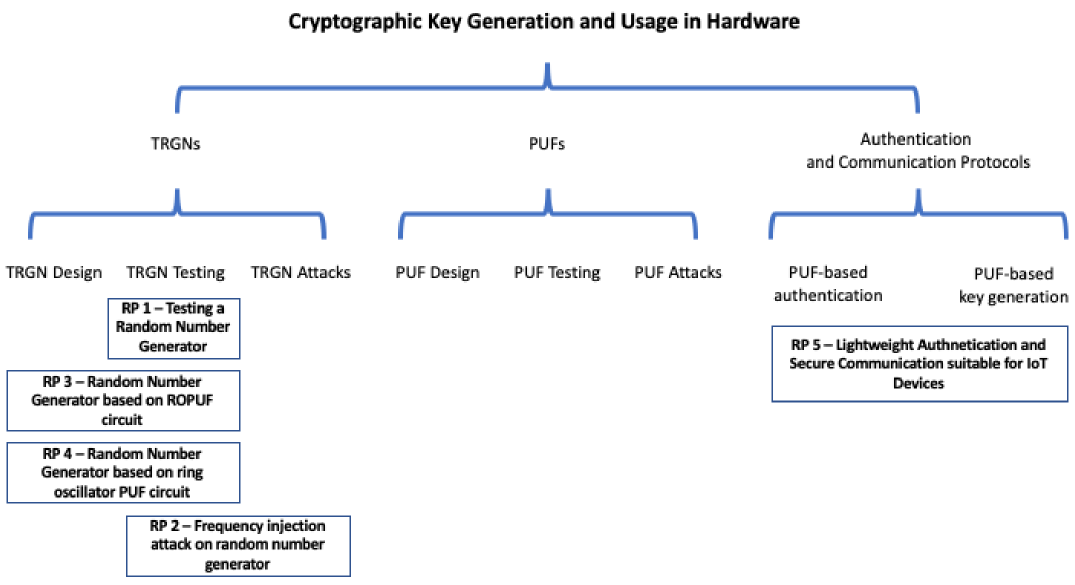

Cryptographic Key Generation and Usage in Hardware
TRGNs
PUFs
Authentication and Communication Protocols
TRGN Design
TRGN Testing
TRGN Attacks
PUF Design
PUF Testing
PUF Attacks
PUF-based authentication
PUF-based key generation
RP 1 – Testing a Random Number Generator
RP 3 – Random Number Generator based on ROPUF circuit
RP 4 – Random Number Generator based on ring oscillator PUF circuit
RP 2 – Frequency injection attack on random number generator
RP 5 – Lightweight Authnetication and Secure Communication suitable for IoT Devices

Figure 4.1: Relationships of author's relevant papers to the thesis topic.

The following relevant papers are included:

RP1 [A.1] *Testing a Random Number Generator.* The paper presents the results of testing a true random generator implemented on an Atmel AVR Microcontroller.

RP2 [A.2] *Frequency injection attack on random number generator.* The paper deals with the attacks on a random number generators and thus importance of testing of generated sequence. The work on papers RP1 and RP2 started within my diploma thesis, however, they pose a solid base for further work performed within the dissertation thesis, thus they are listed for completness.

RP3 [A.3] *True Random Number Generator based on ROPUF circuit.* In this paper the method of generating true random numbers utilizing the circuit primarily designed as PUF based on ring oscillators was proposed.

RP4 [A.5] *True random number generator based on ring oscillator PUF circuit.* The paper follows the work referenced above, especially extending the testing of the proposed TRNG desgin and stability in changing environment.

RP5 [A.6] *Lightweight Authentication and Secure Communication Suitable for IoT Devices.* The paper presents the protocols for lightweight authentication and secure communication for IoT and embedded devices.

4.1 RP1 - Testing a Random Number Generator

When generating the random number bitstream, it's critical to pick up only bits that have sufficient entropy and perform well in statistical tests. Paper RP1 deals with the testing of the generator based on jitter of the RC oscillator with the goal to select such bits. First, the Discrete Fourier Transform was applied to generated sequence to identify component frequencies and to visualize the periodogram of the generated random bit stream. Further, ENT and Diehard test suites were used to identify and pick truly random bits with sufficient entropy, which are secure to use for cryptographic applications. We found out that the generated sequence is not influenced by parasitic frequencies and it does not contain periodically repeating peaks. However, not all the bits of the 16-bit sample are random enough for cryptographic use. We have identified a bit that has sufficient entropy and passes statistical tests, which allowed us to obtain 128 random bits per second. The paper was presented on *POSTER 2011 - 15th International Student Conference on Electrical Engineering* [A.1].

Testing a Random Number Generator

Simona BUCHOVECKÁ[1]

[1] Faculty of Information Technology, Czech Technical University in Prague, Kolejní 550/2, 166 00 Praha, Czech Republic

buchosim@fit.cvut.cz

Abstract. *In this paper we present the testing of a true random generator implemented on an Atmel AVR Microcontroller. This generator uses the jitter of the RC oscillator as the source of entropy. We applied Discrete Fourier Transform to identify component frequencies and to visualize the periodogram of the generated random bit stream. Then we used the ENT and Diehard test suites to identify and pick truly random bits with sufficient entropy, which are secure to use. Finally, we discuss the results and suggest other possible tests.*

Keywords

True Random Number Generator, entropy, Diehard, ENT, Atmel AVR

1. Introduction

Random number generators are used for a large spectrum of applications, e.g. simulations of stochastic systems, numerical analysis, probabilistic algorithms, computer games, cryptography, etc.

For some applications, it is sufficient to use software generated pseudo-random numbers, which have good statistical properties but are not truly random and could be predicted. However, there are applications such as cryptography, where generated numbers have to be truly random and unpredictable. These numbers are used as nonces, one-time pads, salts, or for key generation.

Such a true random number generator is proposed in [1]. It is implemented on an Atmel AVR Microcontroller and it uses the jitter of its built-in RC oscillator as the source of entropy. In this paper we present the results of statistical tests applied to this generator.

2. Random Number Generation

The method of generating random numbers, as presented in [1], utilizes the ATmega169 microcontroller on the AVR Butterfly demo board. The principle of operation is depicted in Fig. 1.

The microcontroller uses its on-chip RC oscillator for the system clock, and an oscillator with an externally connected crystal as an asynchronous clock generator for the Timer/Counter 2 unit. These oscillators are never perfectly stable. Their frequencies are influenced by many physical factors, such as the supply voltage or ambient temperature, and are also subject to inherent jitter.

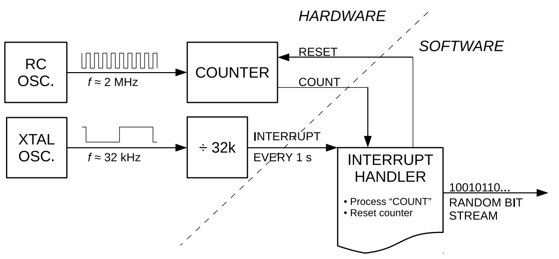


Fig. 1. Principle of random number generation.

The crystal oscillator is much more stable in the entire range of operating conditions, while the RC oscillator, although calibrated, exhibits significant variation. Even if the board is kept in a stable environment and powered form a stabilized source, the RC oscillator exhibits easily measurable jitter.

To generate the random bits, the following method is used. The Timer/Counter 1 unit counts the cycles of the system RC oscillator. The Timer/Counter 2 unit with the low frequency crystal oscillator (32768Hz) and an appropriate prescaler configuration is used to periodically generate an interrupt. In the interrupt handling routine, the 16-bit value in the Timer/Counter 1 register TCNT1 is read (that is, the number of clock cycles mod 2^{16}) and the TCNT1 register is reset to zero.

3. Tests

For our purposes, we used a modification of the RNG with the frequency of the RC oscillator set to 8 MHz. This variant generates 128 16-bit samples per second. We examined a bit stream of approximate length 136 MB.

Some testing had already been done in [2], but this testing suffered from lack of data, and had not been done in detail.

First of all, we examined the raw unmodified data. We presumed there is a possibility of component frequencies in the generated bit stream, so we applied discrete Fourier transform on the sequence of numbers. We looked for frequencies up to approximately 128 Hz, because we presumed that one of the most striking perturbing frequencies would be 50 Hz from the electricity distribution network. The resulting periodogram can be seen in Fig. 2. and in more detail in Fig 3. It is clear, that there are no component frequencies that would influence our generated sequence. The peaks at 0 and at maximal frequency seem to be just artifacts of the used method; they do not appear when examining a wider frequency range.

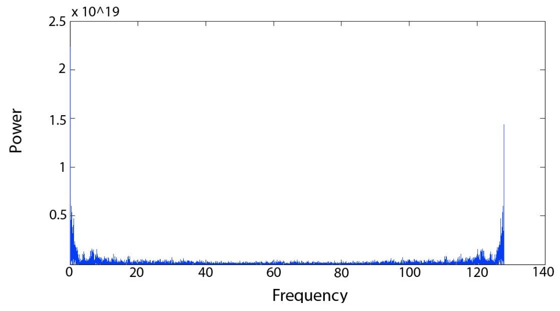


Fig. 2. Periodogram.

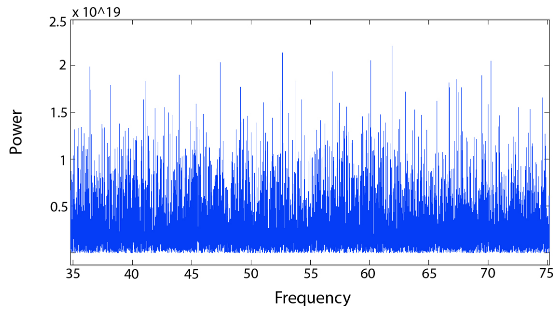


Fig. 3. Detail of periodogram.

Another test we used for detection of possible periodically repeating peaks is based on a simple idea. We picked every *n*-th number (for *n* = 2 to 512), computed the arithmetic mean of such sequence, and compared this arithmetic mean with the arithmetic mean of the complete, unmodified sequence. If there are any periodically repeating peaks, then some of the arithmetic means will be higher than the mean of the whole sequence. Our tests did not show any deviations from expected values.

Considering the results of Discrete Fourier Transform and our mean test, we are able to say that there are no detectable component frequencies and no periodically repeating peaks.

To determine which bits of the generated 16-bit samples have sufficient entropy and are secure to use, we created 16 separate bit streams, *n-th* bit stream consisting of *n-th* bit from each 16-bit sample. Then we tested the entropy and randomness of each of the bit streams.

As stated in [1], the least significant bit (bit 0, LSB), is non-random, and should not be used. It is due to the method used to generate random samples. Before an interrupt can be serviced, the current instruction must be completed. Most AVR instructions take 1 or 2 clock cycles to complete, making LSB highly dependent on the code being executed. In an extreme case, when the code does nothing but wait for interrupts in an infinite loop, the microcontroller only executes RJMP instructions, each of which takes 2 clock cycles, and the LSB stays constant.

First of all we examined the entropy of the individual bit streams. For this purpose, we used the entropy test from the ENT test suite [4]. Only the first bit exhibited sufficient entropy (approximately 7.99 bits per byte). Each subsequent bit had an entropy smaller approximately by one. Entropy of bits 1 to 7 can be seen in Tab. 1., entropies of higher bits were so small that it made no sense to consider them.

Bit	Entropy
1	7.999902
2	6.999952
3	5.999973
4	4.999991
5	3.998549
6	2.854804
7	1.494460

Tab. 1. Entropy of bits 1 to 7

Already this first entropy test made us suspect that only the first bit will be secure to use, and the other bits should be discarded. However, we still applied more tests.

The other applied tests from the ENT test suite were the compression test (based on assumption that a true random number sequence cannot be compressed), the chi-square test, the arithmetic mean of data bytes, the Monte Carlo value for π and the serial correlation coefficient. Results of these tests were similar to the result of the entropy test. The higher bit, the worse result. Bit 1 was the only bit that achieved perfect results.

Since all the bits except bit 1 had insufficient entropy and failed in other tests, we continued testing only with bit stream consisting of the 1st bit of each sample. We applied the Diehard test suite [3]. This bit stream passed all the tests except the OPERM5 test; the result of this test was NaN. This was due to the fact that the input file was not big enough (Diehard test suite requires input files of about 10-12 MB, our bitstream had approximately 8.5 MB).

4. Suggestions for Future Work

We tested the generator operating under normal conditions. However, it is necessary to test the impact of non-standard conditions that could be used for possible attacks. Also, it would be useful to prepare on-line tests, which could immediately detect deviations from randomness and decrease of entropy.

Also, it is possible that by merging the 2nd and the 3rd bit, we will be able to gain one more bit with sufficient entropy. Nevertheless, it is necessary to test the bit stream obtained in this way carefully because there is a possibility that it will not qualify for cryptographic applications.

5. Conclusion

We presented the results of testing a true random generator implemented on an Atmel AVR Microcontroller. We found out that the generated sequence is not influenced by parasitic frequencies and it does not contain periodically repeating peaks. However, not all the bits of the 16-bit sample are random enough for cryptographic use. Only the first bit has sufficient entropy and passes statistical tests. Thus, using this method of generating random numbers, we can obtain 128 random bits per second.

Acknowledgements

Research described in the paper was supervised by Ing. Josef Hlaváč, Ph.D., Faculty of Information Technology, CTU in Prague.

References

[1] HLAVÁČ, J., HADÁČEK, M., LÓRENCZ, R. True Random Number Generation on an Atmel AVR Microcontroller. *Proceedings of 2nd International Conference on Computer Engineering and Technology – ICCET 2010,* 2010, vol. 2, p. 493-495.

[2] HADÁČEK, M. *Generování náhodných čísel na mikrokontrolérech AVR*, bachelor thesis, FEL CTU Prague, 2010.

[3] MARSAGLIA, G. *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness* [online], Florida State University, 1995. Available at http://www.stat.fsu.edu/pub/diehard/.

[4] WALKER, J. *ENT – A Pseudorandom Number Sequence Test Program* [online]. Fourmilab, 2008, Available at http://www.fourmilab.ch/random/.

About Authors...

Simona BUCHOVECKÁ studies in the 1st year of the Informatics master degree program at the Faculty of Information Technology. In her study and research, she focuses on computer security.

4.2 RP2 - Frequency injection attack on random number generator

Various systems are utilizing TRNGs as one of basic building blocks for further usage in multiple cryptographic algorithms. However, compromise of TRNG may lead to compromise of the overall system. Paper RP2 presents a frequency injection attack to show how such compromise of TRNG can be realized. Two variants of the attack are attempted: an invasive attack with a direct modification of the power supply, and a non-invasive attack with no modification of the device. The invasive variant of the frequency injection attack is powerful enough to influence and reduce the randomness of all generated bits – even bit 1 which was found to be truly random in previous work. However, it may be difficult to gain access to the device to connect the parasitic frequency to its power supply line. The non-invasive variant of the attack is not as powerful because it only affected the higher bits of the generated values. Still, significantly less unique values were generated under the attack. One should also consider the fact that no physical manipulation with the device was necessary and no physical traces of the attack were left when it ceased. The results also demonstrate the importance of continuous self-testing of random number generators. In this way, such attacks can be immediately discovered and the random number generator failure properly handled. The paper was presented on *DDECS 2013 - IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits and Systems* [A.2].

Frequency Injection Attack on a Random Number Generator

Simona Buchovecká, Josef Hlaváč
Faculty of Information Technology
Czech Technical University in Prague
buchosim@fit.cvut.cz, josef.hlavac@fit.cvut.cz

***Abstract*—In this paper we present a frequency injection attack on a random number generator implemented in an Atmel AVR microcontroller. Two variants of the attack are attempted: an invasive attack with a direct modification of the power supply, and a non-invasive attack with no modification of the device.**

Keywords—Random Number Generator, Frequency Injection Attack, Atmel AVR

I. Introduction

Random numbers are indispensable in cryptography – they are needed in almost all cryptographic protocols. Random numbers are used to generate public and private keys, nonces (salts), padding, etc. Therefore, the security of the entire cryptosystem also depends on the security of its random number generator. For that reason, a random number generator for cryptographic applications must produce truly random and unpredictable sequences, and must be secured against potential attacks.

Such a true random number generator is proposed in [1]. It is implemented in a low-cost Atmel AVR microcontroller. The jitter of the microcontroller's built-in RC oscillator is used as the source of entropy. In this paper, we present two variants of the frequency injection attack that can affect the randomness of the produced bit sequence and thus reduce the strength of any cryptosystem that would rely on the generator.

II. Random Number Generator

The method of generating random numbers, as presented in [1], utilizes the ATmega169 microcontroller on the AVR Butterfly demo board. The principle is depicted in Fig. 1.

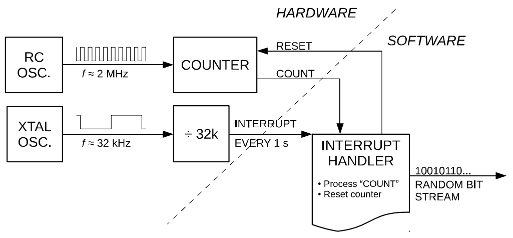


Fig. 1. Principle of random number generation

The microcontroller uses its on-chip RC oscillator for the system clock, and an oscillator with an externally connected crystal as an asynchronous clock generator for the Timer/Counter 2 unit. These oscillators are never perfectly stable. Their frequencies are influenced by many physical factors, such as the supply voltage or ambient temperature, and are also subject to inherent jitter. Due to these factors, the "COUNT" value obtained at every interrupt is different and can be used as a source of entropy.

III. Frequency Injection Attack

A frequency injection attack, which is able to destroy the source of entropy in ring-oscillator-based true random number generators, was proposed in [3]. The main idea is that an on-chip ring oscillator may have a tendency to "synchronize" itself to an externally applied frequency. A contactless variant of the attack is elaborated in [4]; the authors again attempted to influence a ring-oscillator-based true random number generator. We based our attacks on these ideas.

Because the method of generating random numbers described above uses the instability of the RC oscillator as the source of entropy, the goal of our attack is to "stabilize" the oscillator and thus reduce its quality as an entropy source. The reasoning is that by using a relatively stable oscillator operating near the nominal operating frequency (8 MHz) of the RC oscillator, said RC oscillator would lock to this externally applied, parasitic frequency. Consequently, there would be less variation in the obtained "COUNT" values.

In subsequent processing, individual bits from the obtained "COUNT" values are processed. Bit 0 is always thrown away because it is program-dependent. Bit 1 was found to be sufficiently random [5,6] and is therefore added to the output sequence. Other bits, even though they still carry some randomness, are also thrown away.

A. Invasive Variant of the Frequency Injection Attack

In the invasive variant of the attack, a direct access to the device and its power source are needed. We attempted to inject a frequency of 8 MHz from a relatively stable source directly into the device's power supply connection. The principle of the attack is shown in Fig. 2.

The injected frequency is generated by a simple 8 MHz crystal oscillator consisting of a crystal resonator (30 ppm stability) and inverting Schmitt triggers (74HC14). The oscillator is then connected to an external power supply of the AVR Butterfly board as shown in Fig. 2.

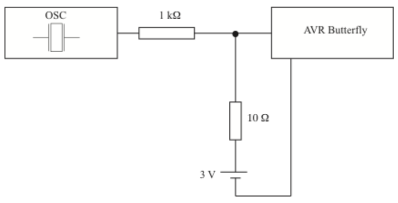


Fig. 2. Modified external power supply circuit

When examining the generated values, we found out that under normal operation with unmodified power supply, 6493 unique "COUNT" values were generated. With the modified power supply and 8 MHz external oscillator, only 2167 unique "COUNT" values appeared in the generated sequence of the same length. The histogram is shown in Fig. 3.

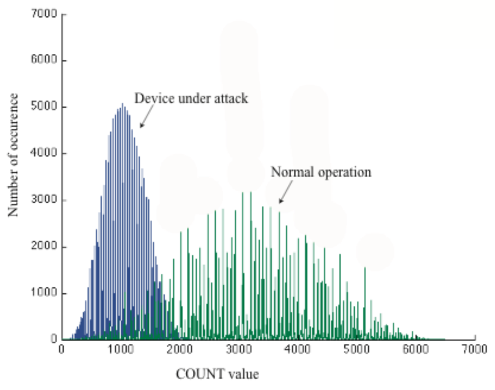


Fig. 3. Frequency injection with modified power supply – histograms. Individual "COUNT" values are plotted on the horizontal axis, the number of occurrences of each respective value on the vertical axis. Wider and lower curve is better and indicates more randomness.

Afterwards, we investigated the randomness of individual bits of the generated numbers using the ENT test suite [2]. We found out that the entropy in the sequence generated under the attack was significantly lower in highest two bits, and the other bits were slightly influenced compared to normal operation, too. Furthermore, results of other tests were much more interesting and revealed attack more clearly.

From previous work [5,6] we know that bit 1 of each generated value is truly random and secure for use even in sensitive applications. However, with the attack, we succeeded in influencing this bit. This bit (as well as all other bits) totally failed in the chi-square test, and also performed worse in other tests (Arithmetic Mean, Monte Carlo, Correlation Coefficient). The effect of the attack is particularly visible in the Monte Carlo test (probabilistic computation of π). The most interesting results are shown in Table I.

TABLE I. INVASIVE FREQUENCY INJECTION

Bit	*Normal operation*	*Device under attack*
	Entropy Test (ideal result 8.0)	
1	7.999902	7.990001
2	6.999952	6.973664
3	5.999973	5.993737
4	4.999991	4.855026
5	3.998549	3.340969
6	2.854804	0.985796
7	1.494460	0.240174
	Arithmetic Mean (ideal result 127.5)	
1	127.4328	132.6571
2	126.8846	117.9227
3	125.9265	118.7512
4	123.9553	123.9264
5	120.7563	119.4672
6	108.9511	17.8832
7	53.6217	188.2342
	Monte Carlo Value for π (Error in %)	
1	3.143755119 (0.07 %)	3.020572084 (3.85 %)
2	3.158404304 (0.54 %)	3.360052455 (6.95 %)
3	3.186437518 (1.43 %)	3.085156954 (1.80 %)
4	3.242756977 (3.22 %)	3.037292025 (3.32 %)
5	3.302272621 (5.11 %)	2.547496107 (18.91 %)
6	3.210195833 (2.18 %)	3.935579051 (25.27 %)
7	3.617043661 (15.13 %)	0.209163183 (93.43 %)

B. Non-Invasive Variant of the Frequency Injection Attack

Because gaining direct physical access to the device and modifying it may not always be possible, we tried a non-invasive variant of the attack, too. In this variant of the attack, a HP 3310A function generator was used to generate an 8 MHz sinusoid signal. A simple antenna was connected to the generator and placed near the AVR Butterfly board with the random number generator running. The generated sequence was captured and analyzed. The principle of the attack is depicted in Fig. 4.

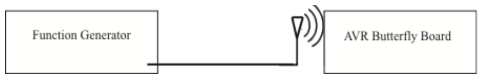


Fig. 4. Non-invasive frequency injection attack – principle

The generated numbers were again visualized in histograms that are shown in Fig. 5. The figure demonstrates that when the device is under the attack, the histogram is significantly thinner and higher. This indicates that fewer unique "COUNT" values were generated and thus less randomness was available.

We again used the ENT test suite [2] to examine the entropy of individual bits in the generated sequence. The generator was not influenced as significantly as in the invasive attack. The influence was evident in the entropy of the two highest bits – it was considerably smaller (2.154833 for bit 6 and 1.30974 for bit 7 when no fault was injected, compared to 0.743732 for bit 6 and 0.351782 when device was under the attack).

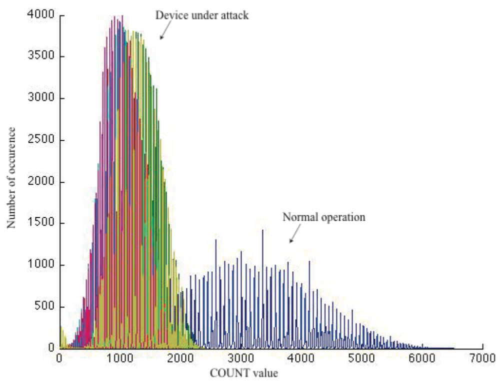


Fig. 5. Non-invasive frequency injection attack – histograms. Multiple measurements with similar results are shown overlapped in the "Device under attack" portion, hence the multiple peaks. Again, individual "COUNT" values are plotted on the horizontal axis, and the number of occurrences of each respective value on the vertical axis.

We also tried to inject a signal that is a harmonic of the RC oscillator operating frequency (8 MHz) – 16, 32 and 48 MHz. The 7th bit was still significantly influenced but the influence on the 6th bit was not so pronounced. The remaining bits were not affected significantly. This may be caused, besides other factors, by the fact that the frequency of the function generator was not as stable compared to the crystal oscillator used in the invasive variant of our attack.

IV. Conclusions

We presented an invasive and a non-invasive variant of a frequency injection attack on a random number generator implemented on an Atmel AVR microcontroller.

The invasive variant of the frequency injection attack is powerful enough to influence and reduce the randomness of all generated bits – even bit 1 which was found to be truly random in previous work [5,6]. However, it may be difficult to gain access to the device to connect the parasitic frequency to its power supply line.

The non-invasive variant of the attack is not as powerful because it only affected the higher bits of the generated values. Still, significantly less unique values were generated under the attack. One should also consider the fact that no physical manipulation with the device was necessary and no physical traces of the attack were left when it ceased.

The results also demonstrate the importance of continuous self-testing of random number generators. In this way, such attacks can be immediately discovered and the random number generator failure properly handled.

Acknowledgment

This work was supported by the Czech Technical University grant No. SGS13/102/OHK3/1T/18 "Information Systems and their Security".

References

[1] J. Hlaváč, M. Hadáček, R.Lórencz, "True random number generation on an Atmel AVR microcontroller," in Proceedings of 2nd International Conference on Computer Engineering and Technology – ICCET 2010, 2010, vol. 2, p. 493-495.

[2] J. Walker, ENT – A pseudorandom number sequence test program [online], Fourmilab 2008, available at http://www.fourmilab.ch/random.

[3] T. Markettos, 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 (CHES 2009), September 2009, p. 317–331.

[4] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, P. Maurine, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in Proceedings of the 3rd International Conference on Constructive Side-Channel Analysis and Secure Design (COSADE'12), May 2012, p. 151–166.

[5] J. Hlaváč, S. Buchovecká, R. Lórencz, "New results in generating true random numbers on simple microcontrollers," in Proceedings of the 9th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi), June 2011.

[6] S. Buchovecká, "Testing a random number generator," in Proceedings of the 15th International Student Conference on Electrical Engineering (POSTER 2011), May 2011.

4.3 RP3 - True Random Number Generator based on ROPUF circuit

The goal of RP3 paper is to prove that it is possible to design the universal crypto system, that can be used for various applications by allowing to generated PUF and TRNG at the same time. The PUF can be utilized for asymmetric cryptography and generating asymmetric keys, TRNG for symmetric cryptography (generating session and ephemeral keys), nonces and salts. In the paper the results of evaluation of such a circuit utilized for TRNG purpose are presented. The presented design is based on 150 pairs of ROs, whose oscillations are counted by two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing. We showed that it is possible to gain up to 142x3 random bits in one run of the ROPUF, but further post processing is needed, which shortens the generated bit stream by 50 % (XOR corrector), or by approx. 75 % (von Neumann corrector). The paper was presented on *DSD 2016 – Euromicro Conference on Digital System Design* [A.3].

2016 Euromicro Conference on Digital System Design

True Random Number Generator based on ROPUF circuit

Simona Buchovecká, Róbert Lórencz, Filip Kodýtek, Jiří Buček
Czech Technical University in Prague
Faculty of Information Technology, Department of Computer Systems
Prague, Czech Republic
simona.buchovecka@fit.cvut.cz, lorencz@fit.cvut.cz, kodytfil@fit.cvut.cz, jiri.bucek@fit.cvut.cz

Abstract— **In this paper we propose the method of generating true random numbers utilizing the circuit primarily designed as PUF based on ring oscillators. The goal is to prove that it is possible to design the universal crypto system, that can be used for various applications - the PUF can be utilized for asymmetric cryptography and generating asymmetric keys, TRNG for symmetric cryptography (generating session and ephemeral keys), nonces and salts. In the paper the results of evaluation of such a circuit utilized for TRNG purpose are presented.**

Keywords—RNG, TRNG, Ring Oscillator, PUF

I. Introduction

In this paper we propose utilization of PUF circuit based on Ring Oscillators [1, 2] as an entropy source for random number generation.

Physically Unclonable Function (PUF) is a function based on physical properties, which are unique for each device. It uses local mismatches and differences between physical components of a device arising during the manufacturing process to generate unpredictable outputs. As such, PUF is suitable for device identification and can be even used for generating and secure storage of private key in asymmetric cryptography – the key is not stored in the volatile memory of the device and cannot be copied out of the device; and as the term "Physical Unclonable Function" indicates, cannot be cloned to the other device.

However, for proper implementation of various cryptographic protocols, not only the ability to generate the asymmetric keys is needed, but also random number generation is crucial. The method for generating random numbers is proposed in this paper.

This paper is organized as follows – this section provides discussion on need of Random Number in cryptography and possibilities of their testing. Section II presents the various approaches to TRNG designs, based on Ring Oscillators, while in Section III our proposal of TRNG is presented. Sections IV and V presents experimental results of proposed TRNG testing, and post processing generated bitstream.

A. Random Number Generators

Different cryptographic protocols and algorithms rely on random numbers. The applications include generating session and ephemeral keys, paddings, nonces and salts etc. Therefore, the quality of random number generator has a significant influence on security of whole cryptosystem. Improperly implemented RNG often leads to compromise of the whole system or reduces the complexity of the attack.

Highly insecure RNG used on Mifare Classic tags decreases the complexity of the attack, that aims to reveal smart card's secret key, by allowing an attacker to precompute a codebook [3].

Sony PS3 protection was broken, because of parameter that should have been random and unique for each ECDSA signature computation, was not randomized properly; thus, allowed attackers to compute private key [4].

Defective random number generators implemented in various ATMs and Point-of-Sale terminals, that are often just counters, allows attackers to harvest authentication codes which enable a "clone" of the card to be used in ATMs and elsewhere [5].

B. Random Number Generator Testing

The quality of Random Number Generator has significant influence the security of whole cryptosystem. Thus, the method to evaluate the randomness of generated sequence is needed. As discussed in [8], randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability. The likely outcome of statistical tests, when applied to a truly random sequence, is known a priori and can be described in probabilistic terms.

General test procedures for studying random data are described in [6] and includes chi-square test, frequency test, serial test, gap test, partition test, run test etc. However, these tests are not designed for RNG used for cryptography; they are rather more general. Another generally focused battery of tests is presented in [7].

The test suite crafted specifically with cryptographic operations in mind was proposed in [8]. The test suite consists of 15 tests, such as frequency test, runs test, entropy test or discrete Fourier transform test etc.

II. True Random Number Generators based on Ring oscillators

The random numbers can be generated either algorithmically (so called deterministic RNG or pseudo-RNG), or entropy source can be realized in hardware (physical or true RNG - TRNG). There are various approaches, how to get entropy from hardware. Analog components are often used as source of entropy, examples include designs based on sinusoidal voltage source [9], avalanche diode [10] or thermal noise of resistor [11]. Another approaches use physical phenomenon such as quantum effects or radioactive decay [12, 13].

The TRNG we propose in this paper utilizes Ring Oscillators (RO) as an entropy source. The idea to use ring oscillators as the source of an entropy is not new. The principle of TRNG generation based on ring oscillators is discussed in [14] - The delay instability of logic gates (inverters) connected into the ring is seen as a frequency/phase instability (a jitter) of the generated clock. The jitter is extracted in a sampling unit using flip-flops or latches.

Variety of TRNG designs based on RO were proposed. In [15] two ROs are used as clocks for linear feedback shift register (LFSR), and a cellular automata shift register (CASR) and output is sampled when a new random number is requested. Design based on Fibonacci and Galois Ring Oscillators is presented in [16]. The randomness as well as robustness can be further increased by XOR-ing the outputs of two oscillators, one being in the Galois and the other in the Fibonacci configuration. The design proposed in [17] consists of two independent and identically configured ring oscillators, a sampling circuit, and a control circuit. The sampler unit uses one clock signal to sample the other clock signal. The stream of samples consists of a run of ones and a gap of zeros. The length of this run and gap is counted modulo 2 and output as a random bit. Oscillator based RNG design is even suitable to be integrated in a Smart Card microcontroller [18].

III. True Random Number Generator Proposal

The TRNG design we propose is based on ROPUF circuit presented in [1, 2]. The principle is depicted in Fig. 1 - the basic building element of the proposed ROPUF/TRNG design is a five stage RO (1 NAND, 4 inverters). Instead of measuring frequency of each RO using reference clock we choose one RO pair and count their oscillations simultaneously using two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing.

Obtained measured counter values are used as output and they are not modified in any way. The work in [1,2] shows that these values are represented in binary code, the appropriately selected part of each binary number for PUF output. Further, the [1, 2] assumes that if we make multiple measurements of one RO pair, bits that are close to the least significant bit (LSB) will vary a lot for example due to environmental changes. On the contrary, bits close to the most significant bit (MSB) will be stable and the environmental changes will have almost no influence on them. The more we will be close to the MSB, the more stable these bits will be. Therefore, it seems, the least significant bits are suitable for random number generation. The further detailed evaluation of such RNG is discussed in Section IV.

IV. Experimental Results

For the measurements, we implemented a circuit containing 150 ring oscillator pairs described in Section III. The measured values were subject to further tests.

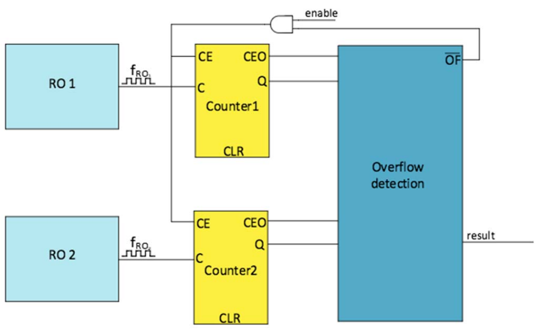


Fig. 1. Principle of operation – basic building block

A. Individual RO Pairs tests

First of all, we examined single bits from every RO pair, as we consider every RO pair as unique source of entropy. As the input sequences for these tests, we used the raw, non post-processed outputs from 1 100 000 measurements. We selected single bit from every RO pair output. Thus, we got 150 x 16 bit streams, as depicted in Fig. 2.

To evaluate randomness, we used the statistical tests in NIST SP 800–22 [8]. The version of the used NIST software is STS 2.1.2. Only some of the tests were run in this setup, as for the tests that require long input bit stream (10^6 bits x 100 sequences at least), we did not have enough data generated. The tests that we were able to run for single bits for each pair of RO were Frequency test (for block length 8), Block frequency test, Cumulative Sums, Runs test, Longest run and Approximate entropy test (for block length 7).

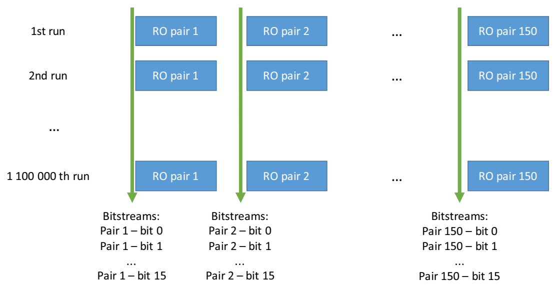


Fig. 2. First test – Individual RO pairs testing

The results of the tests were similar for all of the pairs. Bit 0 fails in some of the tests, bits 1 and 2 shows excellent characteristics, bit 3 fails in some of the tests, and bits 4 and higher fail all of the tests. Therefore, the bits 0 – 3 can be considered for further testing. However, there were some RO pairs, that did not show satisfactory results – only few unique values were generated during multiple runs, resulting the tests to fail, therefore we excluded these pairs from further processing. The results of the tests are summarized in TABLE I. The table shows the resulting values range and average and median values from individual tests of all RO pairs (except excluded ones). The pass rate for the tests is 96/100 and higher.

TABLE I. Test Results – Individual RO tested, average values

TEST	Results			
	Minimal Value	*Maximal value*	*Average*	*Median*
Bit 0				
Frequency	**92/100**	**100/100**	**96,7/100**	**97/100**
BlockFreq.	**96/100**	**100/100**	**99,3/100**	**99/100**
CumSum I	**91/100**	**100/100**	**97,0/100**	**97/100**
CumSum II	**92/100**	**100/100**	**96,8/100**	**97/100**
Runs	**96/100**	**100/100**	**98,9/100**	**99/100**
LongestRun	**96/100**	**100/100**	**99,0/100**	**99/100**
ApproxEntropy	**95/100**	**100/100**	**98,7/100**	**99/100**
Bit 1				
Frequency	**95/100**	**100/100**	**98,6/100**	**99/100**
BlockFreq.	**95/100**	**100/100**	**99,2/100**	**99/100**
CumSum I	**95/100**	**100/100**	**98,6/100**	**99/100**
CumSum II	**95/100**	**100/100**	**98,7/100**	**99/100**
Runs	**96/100**	**100/100**	**99,0/100**	**99/100**
LongestRun	**95/100**	**100/100**	**98,9/100**	**99/100**
ApproxEntropy	**94/100**	**100/100**	**98,7/100**	**99/100**
Bit 2				
Frequency	**95/100**	**100/100**	**98,8/100**	**99/100**
BlockFreq.	**96/100**	**100/100**	**99,2/100**	**99/100**
CumSum I	**95/100**	**100/100**	**98,8/100**	**99/100**
CumSum II	**96/100**	**100/100**	**98,8/100**	**99/100**
Runs	**97/100**	**100/100**	**99,1/100**	**99/100**
LongestRun	**94/100**	**100/100**	**98,9/100**	**99/100**
ApproxEntropy	**96/100**	**100/100**	**98,7/100**	**99/100**
Bit 3				
Frequency	**20/100**	**100/100**	**89,9/100**	**98/100**
BlockFreq.	**0/100**	**100/100**	**78,6/100**	**97/100**
CumSum I	**2/100**	**100/100**	**87,7/100**	**98/100**
CumSum II	**4/100**	**100/100**	**87,9/100**	**98/100**
Runs	**0/100**	**100/100**	**82,2/100**	**97/100**
LongestRun	**19/100**	**100/100**	**92,9/100**	**98/100**
ApproxEntropy	**0/100**	**100/100**	**89,8/100**	**98/100**
Bit 4				
Frequency	**0/100**	**40/100**	**8,1/100**	**7/100**
BlockFreq.	**0/100**	**0/100**	**0,0/100**	**0/100**
CumSum I	**0/100**	**27/100**	**0,8/100**	**0/100**
CumSum II	**0/100**	**30/100**	**0,9/100**	**0/100**
Runs	**0/100**	**7/100**	**0,1/100**	**0/100**
LongestRun	**0/100**	**50/100**	**1,8/100**	**0/100**
ApproxEntropy	**0/100**	**11/100**	**0,2/100**	**0/100**

B. Concatenated RO Pairs outputs tests

The initial tests have shown, that each RO pair can be used as a source of an entropy. Single bits 0 to 3 forming unique bit stream for every RO pair have satisfactory characteristics. However, such setup is not practical for real world use. It makes more sense to concatenate the bits from all of the RO pairs, (as shown in Fig. 3) to reach higher rate of generated bits.

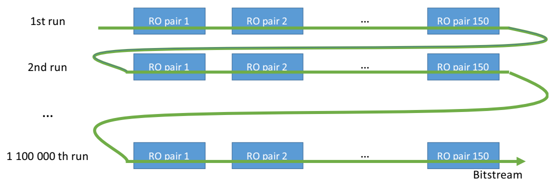


Fig. 3. Concatenating RO pairs outputs

The concatenated bit stream was subject to further tests. After concatenation there was enough data to run all tests from NIST test suite. TABLE II. summarizes the results of testing bits 0 to 3. If the test failed for the distribution of p-Values the cells are highlighted in yellow. The pass rate is 96/100, except for tests Random Excursions and Random Excursions Variants. Further, we also tested concatenated bits 0 to 3 and 1 to 3 into single bit stream. As the bit stream consisting of 0 bits fails not only in frequency and cumulative sums tests, but also in runs test, we examined its influence on combined bit stream. The tests show that the statistical properties of generated bit streams are satisfactory, except for frequency and cumulative sums tests, that are focused on uniform distribution of 0's and 1's in the tested bit stream. However, this often happens, when dealing with truly random number generator. There are various ways of post processing to deal with this problem, if we assume that the bias is the only problem, and the generated bits are statistically independent [19].

TABLE II. Test Results – Concatenated Bit Streams, consisting of bits 0 – 3 from all RO pairs

TEST	Results					
	Bit 0	*Bit 1*	*Bit 2*	*Bit 3*	*Bits 0-3*	*Bits 1-3*
Frequency	**0/100**	**24/100**	**82/100**	**69/100**	**19/100**	**63/100**
BlockFreq.	**99/100**	**98/100**	**99/100**	**98/100**	**100/10 0**	**98/100**
CumSum I	**0/100**	**31/100**	**83/100**	**69/100**	**23/100**	**64/100**
CumSum II	**0/100**	**25/100**	**87/100**	**70/100**	**21/100**	**67/100**
Runs	**0/100**	**83/100**	**95/100**	**94/100**	**69/100**	**91/100**
LongestRun	**99/100**	**99/100**	**98/100**	**100/10 0**	**99/100**	**99/100**
Rank	**100/10 0**	**99/100**	**99/100**	**98/100**	**99/100**	**99/100**
FFT	**97/100**	**96/100**	**99/100**	**100 /100**	**94/100**	**98/100**
NonOverlapping template	**85–100 /100**	**96 -100 /100**	**96 -100 /100**	**96 -100 /100**	**94 -100 /100**	**94 -100 /100**
Overlapping template	**96/100**	**97/100**	**97/100**	**99/100**	**99/100**	**100 /100**
Universal	**97/100**	**98/100**	**97/100**	**98/100**	**100 /100**	**98/100**
ApproxEntropy	**89/100**	**100 /100**	**98/100**	**100 /100**	**98/100**	**99/100**
Random Excursions	**4/4**	**18/18**	**51-52 /52**	**34-35 /35**	**16 -17 /17**	**43- 44 /44**
Random ExcursionVariants	**4/4**	**17-18 /18**	**51-52 /52**	**34-35 /35**	**17/17**	**44/44**
Serial I	**99/100**	**99/100**	**99/100**	**99/100**	**99/100**	**100 /100**
Serial II	**99/100**	**100 /100**	**97/100**	**100 /100**	**99/100**	**99/100**
Linear Complexity	**100 /100**	**99/100**	**98/100**	**99/100**	**100 /100**	**97/100**

V. Post processing of the generated bitstream

To enhance the properties of generated bit stream, we used Von Neumann and XOR post-processing methods.

A. *Post-processing the generated bit stream using Von Neumann corrector*

One of possible methods for de-biasing truly random bit streams was introduced in [20] by Von Neumann. The principle of Von Neumann correction is as follows:

- If the input is "00" or "11", the input is discarded,
- If the input is "10", output a "1",
- If the input is "01", output a "0".

The results of tests after applying the Von Neumann corrector are summarized in TABLE III. and TABLE IV. The bit streams consisting of bits 1-3 shows excellent characteristics after correction. However, it shows, that worse bit 0 characteristics, revealed in previous tests (by failing in runs test) influence the overall bit stream, and thus the bits 1 to 3 can be considered for further use. As the 8 pairs of oscillators were excluded, we are able to get 142 × 3 random bits at one run, shortened by approx. 75% due to post process.

TABLE III. Test Results –Single bits from all RO pairs – Von Neumann correction

TEST	Results			
	Bit 0	*Bit 1*	*Bit 2*	*Bit 3*
Frequency	99/100	100/100	99/100	100/100
BlockFreq.	99/100	100/100	98/100	100/100
CumSum I	100/100	100/100	99/100	100/100
CumSum II	98/100	100/100	99/100	100/100
Runs	97/100	99/100	99/100	98/100
LongestRun	99/100	100/100	98/100	100/100
ApproxEntropy	100/100	98/100	100/100	99/100

TABLE IV. Test Results – Concatenated Bit Streams, consisting of bits 0 – 3 and 1-3 from all RO pairs – Von Neumann Correction

TEST	Results	
	Bits 0-3	*Bits 1-3*
Frequency	69/100	99/100
BlockFreq.	100/100	99/100
CumSum I	69/100	100/100
CumSum II	71/100	99/100
Runs	96/100	99/100
LongestRun	98/100	99/100
Rank	99/100	98/100
FFT	99/100	99/100
NonOverlapping template	98-100/100	97-100/100
Overlapping template	99/100	100/100
Universal	98/100	100/100
ApproxEntropy	100/100	100/100
Random Excursions	30-31/30	62-63/63
Random ExcursionVariants	30-31/30	62-63/63
Serial I	99/100	99/100
Serial II	100/100	100/100
Linear Complexity	100/100	100/100

B. *Post-processing the generated bit stream using XOR corrector*

The results after applying Von Neumann corrector are satisfactory, however the corrector shortens the output stream by 75%. Therefore, we looked at another widely used method – simple XOR correction, which shortens the output stream only by 50%. The XOR corrector takes two subsequent bits from input stream, XORs them and puts the resulting bit into the output stream.

The results of tests after applying the XOR corrector are summarized in TABLE V. and TABLE VI. The results are similar to those after applying Von Neumann corrector – again, bit streams consisting of bits 1-3 shows excellent characteristics after correction. Worse bit 0 characteristics, revealed in previous tests (by failing in runs test) influence the overall bit stream and XOR correction does not solve this issue, and thus the bits 1 to 3 can be considered for further use. As the 8 pairs of oscillators were excluded, at one run of the circuit we are able to get 142 × 3 random bits shortened by 50% due to post process.

TABLE V. Test Results – Single bits from all RO pairs – XOR correction

TEST	Results			
	Bit 0	*Bit 1*	*Bit 2*	*Bit 3*
Frequency	99/100	100/100	99/100	100/100
BlockFreq.	100/100	100/100	100/100	99/100
CumSum I	100/100	99/100	99/100	100/100
CumSum II	100/100	100/100	100/100	99/100
Runs	98/100	99/100	99/100	99/100
LongestRun	97/100	100/100	99/100	98/100
ApproxEntropy	99/100	99/100	100/100	98/100

TABLE VI. Test Results – Concatenated Bit Streams, consisting of bits 0 – 3 and 1-3 from all RO pairs – XOR Correction

TEST	Results	
	Bits 0-3	*Bits 1-3*
Frequency	82/100	98/100
BlockFreq.	99/100	100/100
CumSum I	85/100	99/100
CumSum II	83/100	97/100
Runs	96/100	100/100
LongestRun	99/100	99/100
Rank	99/100	99/100
FFT	100/100	98/100
NonOverlapping template	98-100/100	97-100/100
Overlapping template	99/100	100/100
Universal	99/100	100/100
ApproxEntropy	98/100	100/100
Random Excursions	43-44/100	55/55
Random ExcursionVariants	43-44/100	55/55
Serial I	100/100	100/100
Serial II	98/100	98/100
Linear Complexity	98/100	97/100

VI. Conclusion and Future Work

In this paper we proposed RO based TRNG, utilizing the circuit primarily developed for PUF. We proved, that it is possible to create a universal circuit for generating PUF and TRNG at the same time, for further cryptographic applications. The design is based on 150 pairs of ROs, whose oscillations are counted by two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing.

Up to 142 × 3 random bits can be gained in one run of the ROPUF, but further post processing is needed, which shortens the generated bit stream by 50% (XOR corrector), or by approx. 75% (von Neumann corrector).

Issues of aging or tamper resistance were not addressed in this paper. As stated in [21], the measures that detect defects of this kind in case they occur should be implemented. Three types of tests are distinguished – tot test, startup test and online test. As their names indicate, the tot test shall detect a total breakdown of the noise source, the startup test is used to verify the principle functionality of the noise source when the TRNG has been started, whereas the online test should
detect if the quality of the random numbers is not sufficient or deteriorates in the course of the time. However, the same circuit is utilized for both PUF generation and TRNG, therefore this tests should be addressed and implemented to test both functionalities at once. According to [22], efficient implementation of selected tests from NIST test suite is suitable for on-line monitoring of TRNG. Future work will focus on this issue. The raw, unmodified values read from the counter should be tested on-line to allow evaluation of both TRNG and PUF characteristics over the time.

Acknowledgment

This research has been partially supported by the grant GA16-05179S of the Czech Grant Agency, “Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features” (2016-2018).

References

[1] F. Kodýtek, R. Lórencz, "A Design of Ring Oscillator Based PUF on FPGA," Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on, Belgrade, 2015, pp. 37-42.

[2] F. Kodýtek, R. Lórencz, J. Buček, "Improved ring oscillator PUF on FPGA and its properties", Microprocessors and Microsystems, 2016, *in press.*

[3] K. Nohl, D. Evans, Starbug, and H. Plötz, “Reverse-Engineering a Cryptographic RFID Tag”, In Usenix Sec. Symp., pp. 185-194, 2008.

[4] Failoverflow team: “Console Hacking 2010“, In 27th Chaos Communication Congress, 2010

[5] M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov and R. Anderson, "Chip and Skim: Cloning EMV Cards with the Pre-play Attack," *2014 IEEE Symposium on Security and Privacy*, San Jose, CA, 2014, pp. 49-64.

[6] Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[7] George Marsaglia, The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness (1995)

[8] A. Ruhkin, et al. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication (2010) 800–22 Revision 1a.

[9] N. Sklavos, P. Kitsos, K. Papadomanolakis and O. Koufopavlou, "Random number generator architecture and VLSI implementation," *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. IV-854-IV-857 vol.4.

[10] M. Perić, P. Milićević, Z. Banjac, V. Orlić and S. Milićević, "High speed random number generator for section key generation in encryption devices," *Telecommunications Forum (TELFOR), 2013 21st*, Belgrade, 2013, pp. 117-120.

[11] Huang Zhun and Chen Hongyi, "A truly random number generator based on thermal noise," *ASIC, 2001. Proceedings. 4th International Conference on*, Shanghai, 2001, pp. 862-864.

[12] A. Khanmohammadi, R. Enne, M. Hofbauer and H. Zimmermanna, "A Monolithic Silicon Quantum Random Number Generator Based on Measurement of Photon Detection Time," in *IEEE Photonics Journal*, vol. 7, no. 5, pp. 1-13, Oct. 2015.

[13] F. Xu *et al.*, "A high-speed quantum random number generator prototype," *CLEO: 2013*, San Jose, CA, 2013, pp. 1-2.

[14] B. Valtchanov, A. Aubert, F. Bernard and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in FPGAs," Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on, Bratislava, 2008, pp. 1-6.

[15] T. E. Tkacik, “A hardware random number generator,” in Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '02), B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523 of Lecture Notes in Computer Science, pp. 450–453, Springer, Redwood Shores, Calif, USA, 2003.

[16] J. D. J. Golic, "New Methods for Digital Generation and Postprocessing of Random Data," in IEEE Transactions on Computers, vol. 55, no. 10, pp. 1217-1229, Oct. 2006.

[17] Kohlbrenner, Paul, and Kris Gaj. "An embedded true random number generator for FPGAs." Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM, 2004.

[18] Bucci, Marco, et al. "A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC." Computers, IEEE Transactions on 52.4 (2003): 403-409.

[19] M. Dichtl, "Bad and Good Ways of Post-Processing Biased Physical Random Numbers", in Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, pp.137-152.

[20] J. Von Neumann, “Various Techniques used in Connection with Random Digits”, in National Bureau of Standards Applied Mathematics Series 12, 1951, pp. 36-38.

[21] Werner Schindler and Wolfgang Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. In CHES, pages 431–449, 2002.

[22] Veljković, Filip, Vladimir Rožić, and Ingrid Verbauwhede. "Low-cost implementations of on-the-fly tests for random number generators." Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012. IEEE, 2012.

4.4 RP4 - True random number generator based on ring oscillator PUF circuit

The paper RP4 extends the work performed in the RP3. Additional testing is performed to ensure stability of generated bitstream in changing environment. The set of test ruling out crosstalk and parasitic frequencies was performed. Two experimental setups were verified – one with multiple RO pairs active simultaneously and switching regulator is used as power supply, while another one with setup when single RO pair is active at a given moment and linear regulator is used as power supply. The paper was published in *Microprocessors and Microsystems* [A.5].

Microprocessors and Microsystems 53 (2017) 33–41

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

True random number generator based on ring oscillator PUF circuit

Simona Buchovecká*, Róbert Lórencz, Filip Kodýtek, Jiří Buček

Department of Computer Systems, Faculty of Information Technology, Czech Technical University, Thakurova 9, 160 00 Prague, Czech Republic

ARTICLE INFO

Article history:
Received 9 January 2017
Revised 28 June 2017
Accepted 29 June 2017
Available online 4 July 2017

Keywords:
RNG
TRNG
Ring oscillator
PUF

ABSTRACT

In this paper we propose the method of generating true random numbers utilizing the circuit primarily designed as Physically Unclonable Function (PUF) based on ring oscillators. The goal is to show that it is possible to design the universal crypto system, that can be used for various applications – the PUF can be utilized for asymmetric cryptography and generating asymmetric keys, True Random Number Generator (TRNG) for symmetric cryptography (generating session and ephemeral keys), nonces and salts. In the paper the results of evaluation of such a circuit utilized for TRNG purpose are presented.

1. Introduction

As stated in [5], the security of cryptographic systems is mainly linked to the protection of confidential keys. In high end information security systems, when used in an uncontrolled environment, cryptographic keys should never be generated outside the system and they should never leave the system in clear. For the same reason, if the security system is implemented in a single chip (cryptographic system-on-chip), the keys should be generated inside the same chip.

Implementation of proper methods for cryptographic key generation and their secure storage in logic devices (including configurable logic devices) is of significant importance. Moreover, it is necessary to use the cryptography and corresponding keys in proper manner. Guidelines on how to properly deal with this matter can be found e.g. in NIST Special Publication 800-133 [1].

Nowadays, TRNGs are mostly used for key generation. Therefore, the quality of the random number generator has a significant influence on the security of the whole cryptosystem.

In [7], a radically new approach to secure key storage is defined. With regards to drawbacks of non-volatile storage, authors define the criteria for new approach – authors state that key should not be permanently stored in digital form on the device, it should be uniquely linked to the device, should not be reproducible, and should be only used when required. Thus, the keys should be extracted from the intrinsic properties of the device, and the authors further build their concept for secure key storage on PUFs rather than storing keys in non-volatile memory.

The concept of PUF was originally introduced in [13,14] and the authors showed that instead of relying on number theory, the mesoscopic physics of coherent transport through a disordered medium can be used to allocate and authenticate unique identifiers by physically reducing the medium's microstructure to a fixed-length string of binary digits. These physical one-way functions are inexpensive to fabricate, prohibitively difficult to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. This makes PUF an ideal candidate for providing tamper resistant design for cryptographic key generation and storage.

1.1. Suitability of TRNGs and PUFs for key generation

Strict requirements are laid on the quality of generated cryptographic key for sensitive applications, as well as meeting various mathematical properties of a particular cryptosystem. The key should be unpredictable, and thus is today most often generated using TRNG. Moreover, TRNGs are of great advantage when generating other short-term or public values. TRNGs are suitable for following tasks:

- Key generation for symmetric cryptography – session keys, ephemeral keys,
- Publicly transmitted or open values such as nonces, challenges, salts, initialization vectors,
- Generation of ephemeral keys and other one-time values for asymmetric schemes.

Apart from TRNGs that are traditionally used for generating cryptographic keys, PUFs are being presented as a novel approach increasing physical security. As discussed in [19], safely managing and storing secrets (including cryptographic keys) in mem-

* Corresponding author.
E-mail address: buchosim@fit.cvut.cz (S. Buchovecká).

http://dx.doi.org/10.1016/j.micpro.2017.06.021
0141-9331/© 2017 Elsevier B.V. All rights reserved.

ory is difficult and expensive. Using PUF, the keys are not physically stored, rather they are generated when they are needed. The fact that the key was generated directly on the device and cannot be copied nor cloned, is advantageous especially in the following cases:

- Identification and device authentication,
- Key generation and key storage for symmetric cryptography (for on-chip encryption),
- Key generation and key storage for asymmetric cryptography (private keys).

As such, it is advantageous for proper implementation of various cryptographic protocols to have the ability to generate both random number (e.g. for symmetric keys), as well as PUF (for secure generation and storage of asymmetric keys).

1.2. True random number generators based on ring oscillators

Random numbers can be generated either algorithmically (so called deterministic RNG or pseudo-RNG), or using an entropy source realized in hardware (physical or true RNG – TRNG). There are various approaches, how to get entropy from hardware. Analog components are often used as source of entropy, examples include designs based on sinusoidal voltage source [18], avalanche diode [15] or thermal noise of resistor [24]. Other approaches use physical phenomenon such as quantum effects or radioactive decay [8,23].

The TRNG we propose in this paper utilizes Ring Oscillators (RO) as the entropy source. The idea to use ring oscillators as the source of an entropy is not new. The principle of TRNG generation based on ROs is discussed in [21] – the delay instability of logic gates (inverters) connected into the ring is seen as a frequency/phase instability (a jitter) of the generated clock. The jitter is extracted in a sampling unit using flip-flops or latches.

Variety of TRNG designs based on ROs were proposed. In [20] two ROs are used as clocks for a linear feedback shift register and a cellular automata shift register and output is sampled when a new random number is requested. Design based on Fibonacci and Galois Ring Oscillators is presented in [6]. The randomness as well as robustness can be further increased by XOR-ing the outputs of two oscillators, one being in the Galois and the other in the Fibonacci configuration. The design proposed in [11] consists of two independent and identically configured ROs, a sampling circuit, and a control circuit. The sampling unit uses one clock signal to sample the other clock signal. The stream of samples consists of a run of ones and a gap of zeros. The length of this run and gap is counted modulo 2 and output as a random bit. Oscillator based RNG design is even suitable to be integrated in a Smart Card microcontroller [2].

1.3. Our approach

As discussed in previous sections, from key generation perspective it is advantageous to have the capability to have both PUFs and TRNGs implemented in a single device. Therefore, we propose the method of generating true random numbers utilizing the circuit primarily designed as a PUF based on ROs. We utilize the existing ROPUF circuit [9,10] also for TRNG generation, based on the fact that RO circuits are widely used as entropy sources for TRNGs. In this paper, we verify that it is possible to design the universal crypto system based on ROs, that can be used for various applications – the PUF can be utilized for asymmetric cryptography and generating asymmetric keys, and the TRNG for symmetric cryptography.

The paper is organized as follows – Section 2 presents the proposal of TRNG based on ROPUF circuit. Section 3 provides the results of experimental testing. Two experiments were carried out - during first experimental setup multiple RO pairs are active simultaneously and switching regulator is used as power supply, while during second one only single RO pair is active at a given moment and linear regulator is used as power supply to rule out crosstalk and parasitic frequencies. Section 4 concludes the paper.

2. The proposed true random number generator based on ROPUF circuit

As discussed in previous section, various implementation of cryptographic systems can take an advantage from universal circuit for generating PUF and TRNG at the same time.

In previous work [9,10], the ROPUF circuit was proposed. The principle is depicted in Fig. 1 – the basic building element of the proposed ROPUF design is a five stage RO (1 NAND, 4 inverters). Instead of measuring frequency of each RO using reference clock, we choose two ROs (a RO pair), and count their oscillations simultaneously using two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing. Obtained measured counter values are used as output and they are not modified in any way.

The work in [9,10] shows that these values are represented in binary code, the appropriately selected part of each binary number for PUF output. Further, the [9,10] assumes that if we make multiple measurements of one RO pair, bits that are close to the least significant bit (LSB) will vary a lot (for example, due to environmental changes). On the contrary, bits close to the most significant bit (MSB) will be stable and the environmental changes will have almost no influence on them. The more we will be close to the MSB, the more stable these bits will be.

Given the fact that the entropy of least significant bits of ROPUF circuit is high and bits vary a lot, as well as the RO-based circuits are proven to be secure to be used as TRNG generation, we presented and validated the idea that single RO circuit can be used both for PUF and TRNG generation in [3] – principle is depicted in Fig. 2. The middle bits of generated values (9–7) are used for PUF, while the lowest bits (3–0) with highest entropy are used for TRNG.

3. Experimental results

In following sections, the results of testing of the proposed TRNG are presented. The Digilent Basys 2 prototyping board containing a Xilinx Spartan3E-100 FPGA was used for two separate sets of measurement. For the first measurements (presented in this section), we implemented a circuit containing 150 RO pairs. All RO pairs were active simultaneously during the measurement. An on-board switching regulator was used as the power supply. The scheme of the measurement setup is depicted in Fig. 3. In Section 3.5, we will modify the setup to limit the inherent noise and crosstalk present in the first measurement setup.

The measured values were subject to further tests. To evaluate randomness, we used NIST test suite [16] that was crafted specifically with cryptographic operations in mind. The version of the used NIST software is STS 2.1.2. The test suite consists of 15 tests, such as frequency test, runs test, cumulative sums test, various variations on templates test, entropy test, serial test or discrete Fourier transform test.

3.1. Individual RO pairs tests

First of all, we examined single bits from every RO pair, as we consider every RO pair as a unique source of entropy. As the input sequences for these tests, we used the raw, non post-processed

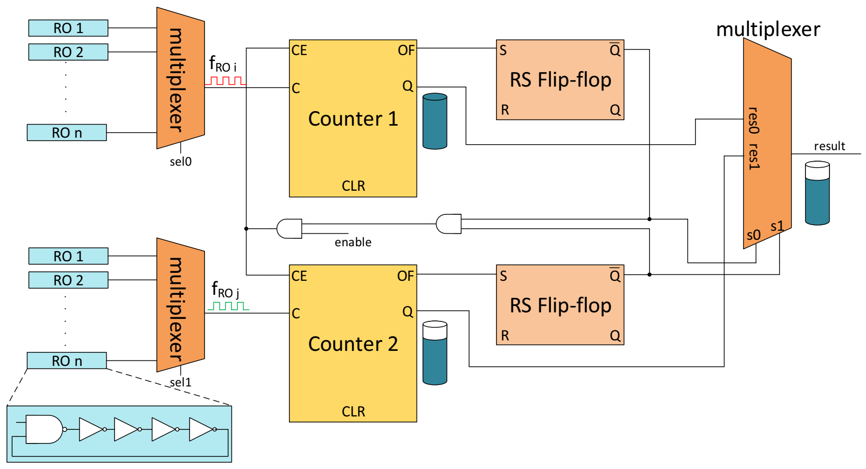


Fig. 1. ROPUF circuit [9,10] used for TRNG generation.

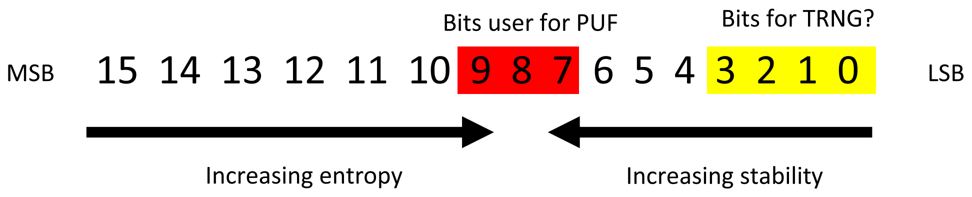


Fig. 2. Behavior of bits in a 16-bit counter – The lower bits, the increasing entropy.

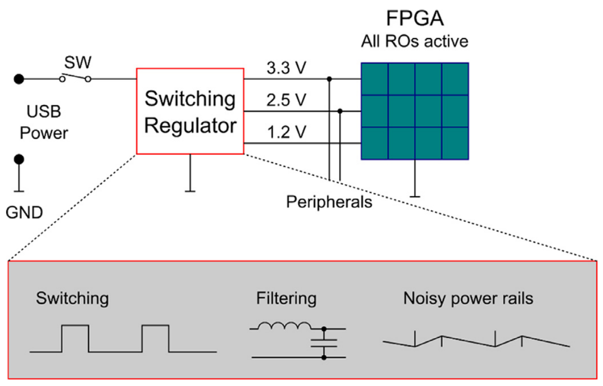


Fig. 3. First experimental setup – all RO pairs active simultaneously, a switching regulator used as power supply.

outputs from 1,100,000 measurements. We selected single bit from every RO pair output. Thus, we got 150×16 bit streams, as depicted in Fig. 4.

Only some of the tests from NIST test suite were run in this setup, as for the tests that require long input bit stream (106 bits × 100 sequences at least), we did not have enough data generated. The tests that we were able to run for single bits for each pair of RO were Frequency test (for block length 8), Block frequency test, Cumulative Sums, Runs test, Longest run and Approximate entropy test (for block length 7).

The results of the tests were similar for all of the pairs. Bit 0 fails in some of the tests, bits 1 and 2 show excellent characteristics, bit 3 fails in some of the tests, and bits 4 and higher fail all of the tests. Therefore, the bits 0 – 3 can be considered for further testing. However, there were some RO pairs, that did not show satisfactory results – only few unique values were generated during multiple runs, resulting the tests to fail, therefore we excluded these pairs from further processing. The results of the tests are summarized in Table 1. The table shows the resulting median values from individual tests of all RO pairs (except excluded ones). The pass rate for the tests is 96/100 and higher.

3.2. Concatenated RO pairs outputs tests

The initial tests have shown, that each RO pair can be used as a source of entropy. Single bits 0 to 3 forming unique bit stream for every RO pair have satisfactory characteristics. However, such setup is not practical for real world use. It makes more sense to concatenate the bits from all of the RO pairs, (as shown in Fig. 5) to reach higher rate of generated bits.

The concatenated bit stream was subject to further tests. After concatenation there was enough data to run all tests from the NIST test suite. Table 2 summarizes the results of testing bits 0 to 3. If the test failed for the distribution of *p*-values the cells are highlighted in bold. The pass rate is 96/100, except for Random Excursions and Random Excursions Variants. Further, we also tested concatenated bits 0 to 3 and 1 to 3 into single bit stream. As the bit stream consisting of 0 bits fails not only in frequency and cumulative sums tests, but also in runs test, we examined its influence on combined bit stream. The tests show that the statistical properties of generated bit streams are satisfactory, except for frequency and cumulative sums tests, that are focused on uniform distribution of 0′s and 1′s in the tested bit stream. However, this often happens when dealing with truly random number generators. There are various ways of post processing to deal with this problem, if we assume that the bias is the only problem, and the generated bits are statistically independent [4]. To enhance the properties of gener-

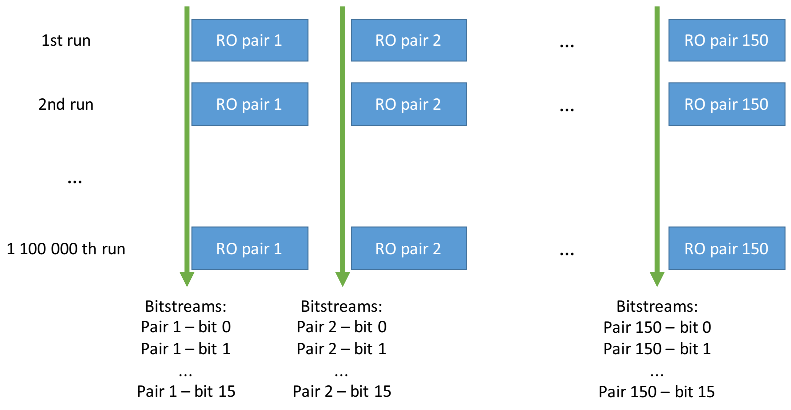


Fig. 4. Initial test - Single bits from every RO pair examined.

Table 1
Individual RO test–results of NIST STS tests. Each test was run for 150 generated bit streams and median and average values from 150 test output values is presented. Minimum allowed pass rate is 96/100.

Bit	0		1		2		3		4	
	Median	Average	Median	Average	Median	Average	Median	Average	Median	Average
Frequency	97/100	96,7/100	99/100	98,6/100	99/100	98,8/100	98/100	89,9/100	7/100	8,1/100
Block frequency	99/100	99,3/100	99/100	99,2/100	99/100	99,2/100	97/100	78,6/100	0/100	0,0/100
Cumulative sums I	97/100	97,0/100	99/100	98,6/100	99/100	98,8/100	98/100	87,7/100	0/100	0,8/100
Cumulative sums II	97/100	96,8/100	99/100	98,7/100	99/100	98,8/100	98/100	87,9/100	0/100	0,9/100
Runs	99/100	98,9/100	99/100	99,0/100	99/100	99,1/100	97/100	82,2/100	0/100	0,1/100
Longest run	99/100	99,0/100	99/100	98,9/100	99/100	98,9/100	98/100	92,9/100	0/100	1,8/100
Approximate entropy	99/100	98,7/100	99/100	98,7/100	99/100	98,7/100	98/100	89,8/100	0/100	0,2/100

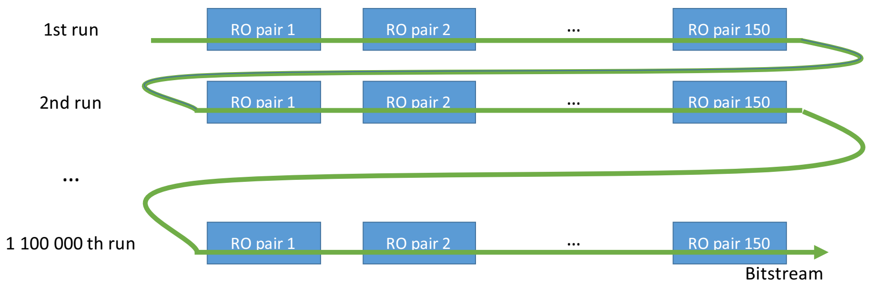


Fig. 5. Concatenating bits across RO pairs into single stream to get higher rate of generated bits.

ated bit stream, we involved post processing methods in further testing.

3.3. *Post-processing the generated bit stream using Von Neumann corrector*

One of possible methods for de-biasing truly random bit streams was introduced by Von Neumann and is discussed in detail in [12]. The principle of Von Neumann correction is as follows, if the input is “00” or “11”, the input is discarded, if the input is “10”, output a “1”, if the input is “01”, output a “0”.

The results of tests after applying the Von Neumann corrector are summarized in Table 3. The bit streams consisting of bits 1–3 shows excellent characteristics after correction. However, it shows, that worse bit 0 characteristics, revealed in previous tests (by failing in runs test) influence the overall bit stream, and thus the bits 1 to 3 can be considered for further use. As the 8 pairs of oscillators were excluded, we are able to get 142×3 random bits at one run, shortened by approx. 75% due to post process using Von Neumann corrector.

3.4. *Post-processing the generated bit stream using XOR corrector*

The results after applying Von Neumann corrector are satisfactory, however the corrector shortens the output stream by 75%. Therefore, we looked at another widely used method – simple XOR correction, which shortens the output stream only by 50%, as it takes two subsequent bits from input stream, XORs them and puts the resulting bit into the output stream.

The results of tests after applying the XOR corrector are summarized in Table 3. The results are similar to those after applying Von Neumann corrector – again, bit streams consisting of bits 1–3 shows excellent characteristics after correction. Worse bit 0 characteristics, revealed in previous tests (by failing in runs test) influence the overall bit stream and XOR correction does not solve this issue, and thus the bits 1 to 3 can be considered for further use. As the 8 pairs of oscillators were excluded, at one run of the circuit we are able to get 142×3 random bits shortened by 50% due to post process.

Table 2
Concatenated RO test–The bit streams from *n*th bit from all RO pairs were concatenated and tested using NIST STS test suite. Furthermore, to achieve higher rate of generated bit stream, also bits 0–3 and 1–3 were concatenated. Minimum allowed pass rate is 96/100. The bold values indicate that the test failed for the distribution of *p*-values.

	Individual bits			Concatenated bits		
Position	0	1	2	3	0-3	1-3
Frequency	**0/100**	**24/100**	**82/100**	**69/100**	**19/100**	**63/100**
Block Frequency	99/100	98/100	99/100	98/100	100/100	98/100
Cumulative Sums I	**0/100**	**31/100**	**83/100**	**69/100**	**23/100**	**64/100**
Cumulative Sums II	**0/100**	**25/100**	**87/100**	**70/100**	**21/100**	**67/100**
Runs	**0/100**	83/100	95/100	94/100	**69/100**	91/100
Longest Run	99/100	99/100	98/100	100/100	99/100	99/100
Rank	100/100	99/100	99/100	98/100	99/100	99/100
FFT	97/100	96/100	99/100	100 /100	94/100	98/100
Non Overlapping template	85-100/100	96 -100/100	96 -100 /100	96 -100 /100	94 -100 /100	94 -100 /100
Overlapping Template	96/100	97/100	97/100	99/100	99/100	100 /100
Universal	97/100	98/100	97/100	98/100	100 /100	98/100
Approximate Entropy	89/100	100 /100	98/100	100 /100	98/100	99/100
Random Excursions	4/4	18/18	51-52 /52	34-35 /35	16 -17/17	43- 44/44
Random Excursions Variants	4/4	17-18 /18	51-52 /52	34-35 /35	17/17	44/44
Serial I	99/100	99/100	99/100	99/100	99/100	100 /100
Serial II	99/100	100 /100	97/100	100 /100	99/100	99/100
Linear Complexity	100 /100	99/100	98/100	99/100	100 /100	97/100

3.5. Ruling out crosstalk and parasitic frequencies

To rule out potential crosstalk between individual RO pairs and parasitic frequencies from switching regulators influencing randomness of generated bit stream and thus to verify that each RO pair can be considered as self-contained entropy source, we tested individual pairs separately, using a set of linear regulators as power supply. For this purpose, a circuit containing 130 RO pairs was implemented. The lower number of RO pairs is caused by additional logic needed to allow each RO pair to be run and measured separately (as opposed to normal operation, when all the RO pairs run and are measured together). The Digilent Basys 2 prototyping board was modified so that the original power supply circuit is disconnected and the new power supply consists of a battery and linear regulators. Each RO pair was run individually 1,100,000 times and the generated values were subject to further tests. The scheme of the experiment setup is depicted in Fig. 6. Same testing approach as described in previous section was applied.

First of all, to confirm that every RO pair is a unique source of entropy, individual bits from every RO pair were examined. As input data for this test we used raw, non post-processed measured outputs. Similarly, as in first test (chapter a), we selected single bit from every RO pair input and we got 130 × 16 bit streams. The results were almost the same for all of the pairs. Bits 0–2 pass the tests successfully and seems to have satisfactory results even when each pair is measured separately. The results are summarized in Table 4. However, there were again twelve RO pairs, that did not show satisfactory results – only few unique values were generated

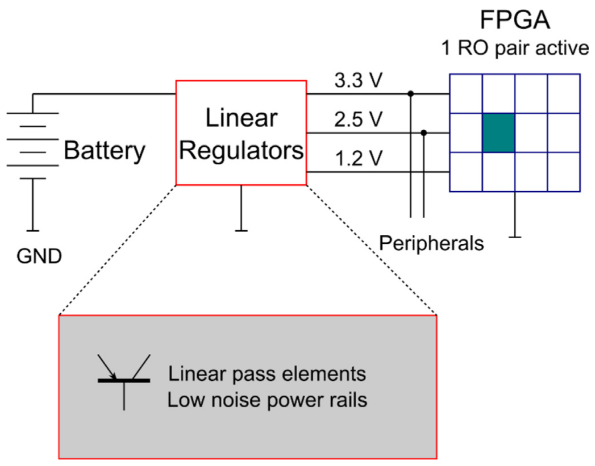


Fig. 6. Second experimental setup – one RO pair active at given moment, linear regulator used as power supply.

during multiple runs, resulting the tests to fail, therefore we excluded these pairs from further processing.

Further, we proceed with same tests as presented in previous Sections 3.2–3.4 – testing concatenated single bits from multiple RO pairs, concatenated multiple bits from multiple RO pairs, finish-

Table 3
Results of NIST STS tests of concatenated bit stream after applying Von Neumann and XOR Correction. Minimum allowed pass rate is 96/100. The bold values indicate that the test failed for the distribution of *p*-values.

	Concatenated bits			
	Von Neumann Correction		XOR Correction	
Position	0-3	1-3	0-3	1-3
Frequency	**69/100**	99/100	**82/100**	98/100
Block Frequency	100/100	99/100	99/100	100/100
Cumulative Sums I	**69/100**	100/100	**85/100**	99/100
Cumulative Sums II	**71/100**	99/100	**83/100**	97/100
Runs	96/100	99/100	96/100	100/100
Longest Run	98/100	99/100	99/100	99/100
Rank	99/100	98/100	99/100	99/100
FFT	99/100	99/100	100/100	98/100
Non Overlapping template	98-100/100	97-100/100	98 -100/100	97-100/100
Overlapping Template	99/100	100/100	99/100	100/100
Universal	98/100	100/100	99/100	100/100
Approximate Entropy	100/100	100/100	98/100	100/100
Random Excursions	30-31/30	62-63/63	43-44/100	55/55
Random Excursions Variants	30-31/30	62-63/63	43-44/100	55/55
Serial I	99/100	99/100	100/100	100/100
Serial II	100/100	100/100	98/100	98/100
Linear Complexity	100/100	100/100	98/100	97/100

Table 4
Individual RO pair test – results of NIST STS tests. Each test was run for 130 generated bit streams and median and average values from 130 test output values is presented. Minimum allowed pass rate is 96/100.

Bit	0		1		2		3	
	Median	Average	Median	Average	Median	Average	Median	Average
Frequency	99/100	89,7/100	99/100	89,7/100	99/100	89,7/100	58/100	58,3/100
Block frequency	99/100	90,6/100	99/100	90,1/100	99/100	90,1/100	0/100	0,7/100
Cumulative sums I	99/100	89,8/100	99/100	89,7/100	99/100	89,7/100	40/100	43,6/100
Cumulative sums II	99/100	89,7/100	99/100	89,7/100	99/100	89,7/100	38/100	43,5/100
Runs	99/100	89,8/100	99/100	89,8/100	98/100	82/100	0/100	1/100
Longest run	99/100	89,9/100	99/100	89,9/100	99/100	89,8/100	0/100	0,2/100
Approximate entropy	99/100	89,9/100	99/100	89,9/100	98/100	87,9/100	0/100	0,1/100

ing with post-processing using XOR and Von Neumann corrector. Final results for concatenated bits 0–2 are summarized in Table 5. The results are very similar in both cases – whether the output is measured from all RO pairs simultaneously or separately. In both cases, 3 bits from each RO pair can be used. Thus, we can presume that each individual RO pair is a unique source of entropy, and the ROPUF design [9,10] can be also securely used for random number generation purposes.

4. Conclusion

Both TRNGs and PUFs have different characteristics that are advantageous in different applications. The paper discussed requirements laid on cryptographic keys in various cryptosystems, and thus suitability of TRNG and PUF for these tasks. It seems that it is most suitable to use TRNG for symmetric cryptography, especially for generating session and ephemeral keys. On the other hand, PUF is advantageous in applications dealing with asymmetric cryptography and storage of private keys.

Based on this, we proposed RO based TRNG, utilizing the circuit primarily developed for PUF. We showed that the design proposed in [9,10] can be used for both PUF and TRNG. The design is based on pairs of ROs, whose oscillations are counted by two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing.

Table 5
Results of NIST STS tests of concatenated bit stream from individually measured ROs without post-process and after applying Von Neumann and XOR Correction. Minimum allowed pass rate is 96/100. The bold values indicate that the test failed for the distribution of *p*-values.

	Concatenated bits 0-2		
Test	Without post-process	XOR corrector	Von Neumann corrector
Frequency	**53/100**	99/100	99/100
Block Frequency	**85/100**	99/100	99/100
Cumulative Sums I	**55/100**	98/100	99/100
Cumulative Sums II	**52/100**	98/100	99/100
Runs	**81/100**	98/100	94/100
Longest Run	97/100	99/100	100/100
Rank	99/100	98/100	99/100
FFT	98/1000	100/100	100/100
Non Overlapping template	85-100/100	95-100/100	97-100/100
Overlapping Template	91/100	99/100	99/100
Universal	98/1000	100/100	100/100
Approximate Entropy	94/100	98/100	95/100
Random Excursions	29-30/30	12/12	7-8/8
Random Excursions Variants	29-30/30	13/13	8/8
Serial I	94/100	100/100	99/100
Serial II	100/100	100/100	99/100
Linear Complexity	100/100	98/100	99/100

Two experimental setups were verified – one with multiple RO pairs active simultaneously and switching regulator is used as power supply, while another one with setup when single RO pair is active at a given moment and linear regulator is used as power supply. Up to three random bits can be gained in one run of the ROPUF from one RO pair in both cases, but further post processing is needed, which shortens the generated bit stream by 50% (XOR corrector), or by approx. 75% (von Neumann corrector).

Issues of aging or tamper resistance of proposed circuit were not addressed in this paper. As stated in [17], the measures that detect defects of this kind in case they occur should be implemented. Three types of tests are distinguished – tot test, start-up test and online test. As their names indicate, the tot test shall detect a total breakdown of the noise source, the start-up test is used to verify the principle functionality of the noise source when the TRNG has been started, whereas the online test should detect if the quality of the random numbers is not sufficient or deteriorates in the course of the time. However, the same circuit is utilized for both PUF generation and TRNG, therefore these tests should be addressed and implemented to test both functionalities at once. According to Veljković et al. [22], efficient implementation of selected tests from NIST test suite is suitable for on-line monitoring of TRNG. Future work will focus on this issue. The raw, unmodified values read from the counter should be tested on-line to allow evaluation of both TRNG and PUF characteristics over the time.

Acknowledgments

This research has been partially supported by the grant GA16-05179S of the Czech Grant Agency, “Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features” (2016–2018).

References

[1] E. Barker, A. Roginsky, Recommendation for cryptographic key generation, NIST Spec. Publ. 800 (2012) 133.
[2] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, M. Varanonuovo, A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC, IEEE Trans. Comput. 52 (4) (2003) 403–409.
[3] S. Buchovecká, F. Kodýtek, R. Lórencz, J. Buček, True random number generator based on ROPUF circuit, in: Digital System Design (DSD), 2016 Euromicro Conference on, IEEE, 2016.
[4] M. Dichtl, Bad and good ways of post-processing biased physical random numbers, in: International Workshop on Fast Software Encryption, Berlin Heidelberg, Springer, 2007.
[5] V. Fischer, A closer look at security in random number generators design, in: International Workshop on Constructive Side-Channel Analysis and Secure Design, Berlin Heidelberg, Springer, 2012.
[6] J.D.J. Golic, New Methods for Digital Generation and Postprocessing of Random Data, IEEE Trans. Comput. 55 (10) (2006) 1217–1229.
[7] H. Handschuh, S. Geert-Jan, T. Pim, Hardware intrinsic security from physically unclonable functions, Towards Hardware-Intrinsic Security, Springer, Berlin Heidelberg, 2010, pp. 39–53.
[8] A. Khanmohammadi, R. Enne, M. Hofbauer, H. Zimmermanna, A monolithic silicon quantum random number generator based on measurement of photon detection time, IEEE Photonics J. 7 (5) (2015) 1–13.
[9] F. Kodýtek, R. Lórencz, A design of ring oscillator based PUF on FPGA, in: Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on, IEEE, 2015.

[10] F. Kodýtek, R. Lórencz, J. Buček, Improved ring oscillator PUF on FPGA and its properties, Microprocess. Microsyst. (2016).
[11] P. Kohlbrenner, Gaj Kris, An embedded true random number generator for FPGAs, in: Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, ACM, 2004.
[12] J. Von Neumann, 13. Various Techniques Used in Connection with Random Digits, J. Res. NBS Appl. Math. Ser. 12 (1951) reprinted in John von Neumann: Collected Works, Vol. V, Pergamon Press, Oxford (1963).
[13] P.S. Ravikanth, Physical One-Way Functions, Diss. Massachusetts Institute of Technology, 2001.
[14] R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, Physical one-way functions, Science 297 (5589) (2002) 2026–2030.
[15] M. Perić, P. Milićević, Z. Banjac, V. Orlić, S. Milićević, High speed random number generator for section key generation in encryption devices, in: Telecommunications Forum (TELFOR), 2013 21st, Belgrade, 2013, pp. 117–120.
[16] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., and Barker, E. A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special Publication (2010) 800–22 Revision 1a.
[17] W. Schindler, W. Killmann, Evaluation criteria for true (physical) random number generators used in cryptographic applications, in: CHES, 2002, pp. 431–449.
[18] N. Sklavos, P. Kitsos, K. Papadomanolakis, O. Koufopavlou, Random number generator architecture and VLSI implementation, in: Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, 2002 pp. IV-854-IV-857 vol.4.
[19] G. Edward Suh, S. Devadas, Physical unclonable functions for device authentication and secret key generation, in: Proceedings of the 44th Annual Design Automation Conference, ACM, 2007.
[20] T.E. Tkacik, A hardware random number generator, in: B.S. Kaliski, Ç.K. Koç, C. Paar (Eds.), Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '02), Redwood Shores, Calif, USA, Springer, 2003, pp. 450–453.
[21] B. Valtchanov, A. Aubert, F. Bernard, V. Fischer, Modeling and observing the jitter in ring oscillators implemented in FPGAs, in: Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on, Bratislava, 2008, pp. 1–6.
[22] F. Veljković, V. Rožić, I. Verbauwhede, Low-cost implementations of on-the-fly tests for random number generators, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, IEEE, 2012.
[23] F. Xu, et al., A high-speed quantum random number generator prototype, in: CLEO: 2013, San Jose, CA, 2013, pp. 1–2.
[24] Z. Huang, H. Chen, A truly random number generator based on thermal noise, in: ASIC, 2001. Proceedings. 4th International Conference on, Shanghai, 2001, pp. 862–864.

4.5 RP5 - Lightweight Authentication and Secure Communication Suitable for IoT Devices

The paper RP5 presents the set of lightweight protocols enabling secure authentication and communication. The protocols are using a PUF/TRNG combined circuit as a basic building block. The goal is to show the possibilities of securing communication and authentication of the embedded systems, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and micro-controllers, while allowing secure communication. For the authentication, several protocols based on pre-generated PUF challenge/response values are proposed. Since the PUF responses are unique per each device and are intrinsically random, PUF is ideal cryptographic primitive for this purpose. Three variants of the protocol are discussed – authentication against central authority using PUF challenge and encrypted response, and two variants of authentication that use the PUF for key generation – single device authentication, and mutual device authentication. After the authentication process, a shared key between the devices is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using conventional or lightweight block ciphers as needed. The paper was presented on *ICISSP 2020 - International Conference on Information Systems Security and Privacy* [A.6].

Lightweight Authentication and Secure Communication Suitable for IoT Devices

Simona Buchovecká, Róbert Lórencz, Jiří Buček and Filip Kodýtek
Department of Information Security, Faculty of Information Technology,
Czech Technical University in Prague, Czech Republic
{simona.buchovecka, lorencz, jiri.bucek, kodytfil}@fit.cvut.cz

Keywords: Authentication, Secure Communication, PUF, TRNG, Key Generation, Key Management, IoT Security.

Abstract: In this paper we present the protocols for lightweight authentication and secure communication for IoT and embedded devices. The protocols are using a PUF/TRNG combined circuit as a basic building block. The goal is to show the possibilities of securing communication and authentication of the embedded systems, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and microcontrollers, while allowing secure communication.

1 INTRODUCTION

Implementation of proper methods for authentication and secure communication, including secure management of key material in lightweight devices, is of significant importance, especially with rise of IoT devices. The use of cryptography and corresponding keys in proper manner is one of the leading problems, when talking about devices with limited computing resources and low power consumption.

Variety of communication protocols were proposed for secure authentication and commuunication in IoT world, such as machine-to-machine/Internet of Things connectivity protocol (MQTT), Constrained Application Protocol (CoAP), or Datagram Transport Layer Security (DTLS) that can be integrated with CoAP. However, those are still rather heavy-weight and computationally quite expensive protocols when considering simple and constrained devices and thus their variants are constantly being present, for instance Lithe (Raza et al., 2013) or E-Lithe (Haroon et al., 2007) as a lightweight variant of DTLS (Tschofenig et al., 2016). Moreover, these protocols do not deal with secure generation and storage of cryptographic keys, which is rather prerequisity for their usage.

Before introducing the principles of Physically Unclonable Functions (PUFs), we shall summarize the currently most widely used methods for key generation and storage. Nowadays, Random Number Generators (RNGs) are mostly used for key generation that are further used in cryptographic protocols for authentication and secure communication. For lightweight and embedded devices, the True Random Number Generators (TRNGs) are usually implemented, utilizing non-deterministic effects in analogue or digital circuits, since this is resource and power efficient way. Therefore, the quality of TRNG has a significant influence on security of whole system. Improperly implemented TRNG often leads to compromise of the whole system or reduces the complexity of the attack.

Moreover, once the key is generated, it needs to be stored securely in the device (Handschuh et al., 2010), e.g. utilizing storage with tamper-resistance techniques implemented. However, to implement such measures is a complex and cost-ineffective task,

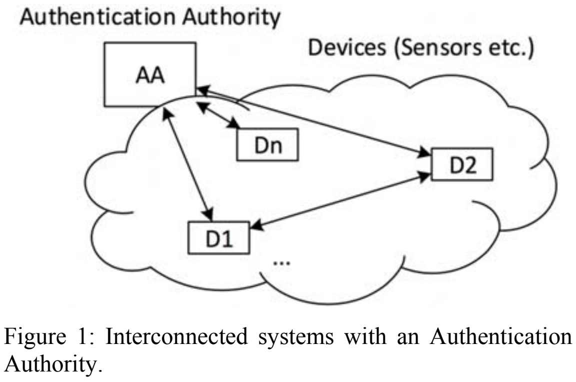


Figure 1: Interconnected systems with an Authentication Authority.

therefore often neglected in practical applications.

Thus, properly defined and implemented key management, including proper key generation, key storage and key usage for various applications (authentication, access control, encryption) in interconnected IoT and embedded systems, as depicted in Figure 1 is still a challenging task (Roman et al., 2013, Malina et al., 2016). A consistent way of handling various cryptographic keys, possibilities of reusing traditional security mechanisms and ensuring end-to-end integrity verification mechanisms is needed (Sicari et al., 2015). All security protocols require credentials, thus optimal key management systems must be implemented to store and distribute these credentials (Roman et al., 2013).

The systematic and formalized approach to key management in IoT devices and embedded systems with properly defined requirements, as well as efficient light-weight modules for key generation, storage and secure usage are missing. However, the need for proper key management in particular applications of embedded systems and IoT started to be raised in some research papers with regards to specific areas such as automotive context (Schleiffer et al., 2013), distributed sensor networks (Chan et al., 2005), or embedded systems in general (Sklavos et al., 2016).

Though TRNGs are mostly used nowadays for cryptographic key generation, in recent years, numerous works dealing with Physical Unclonable Functions (PUFs) for key generation had been published, proposing PUFs as another possible approach for key generation. The concept of PUF was originally introduced in (Pappu, 2002), showing that instead of relying on number theory, the mesoscopic physics of coherent transport through a disordered medium can be used to allocate and authenticate unique identifiers by physically reducing the medium's microstructure to a fixed-length string of binary digits. These physical one-way functions are inexpensive to fabricate, prohibitively difficult to duplicate, admit no compact mathematical representation, and are intrinsically tamper-resistant. This makes PUF as ideal candidate for providing tamper resistant design for cryptographic key generation and storage.

Therefore, PUFs usage is promising to solve the issue of secure storage of cryptographic keys. Instead of storing the key in memory, the key is generated at the time it is needed. A combined PUF/TRNG circuit used in our paper is therefore a suitable alternative for the purpose of key generation and authentication in lightweight cryptographic applications, such as IoT devices and other embedded platforms.

The structure of this paper is as follows. In Section 2, related work is summarized. In Section 3, our proposed approach to lightweight authentication and secure communication is presented. Section 4 presents a case study with a specific PUF/TRNG circuit. Section 5 concludes this paper.

2 RELATED WORK

Every protocol for authentication and secure communication requires cryptographic keys. The minimum common requirements for key generation and storage are summarized by Maes et al. (Maes et al, 2012): A source of true randomness that ensures unpredictable and unique fresh keys; and a protected memory which reliably stores the keys information while shielding it completely from unauthorized parties. As further discussed in (Fischer, 2012), the security of cryptographic systems is mainly linked to the protection of confidential keys. In high-end information security systems, when used in an uncontrolled environment, cryptographic keys should never be generated outside the system and they should never leave the system in clear. For the same reason, if the security system is implemented in a single chip (cryptographic system- on-chip), the keys should be generated inside the same chip.

As mentioned above, for secure key generation a source of true randomness is needed, and the generated keys should be unpredictable. Thus, for proper generation of cryptographic keys random bit stream is required. Therefore, traditional methods of generating cryptographic keys in hardware and embedded systems are mainly based on true random number generators (TRNGs). As stated by Schindler (Schindler, 2009), ideal random number generators are characterized by the property that the generated random numbers are independent and uniformly distributed on a finite range. Various TRNG designs suitable for cryptographic key generation include purely digital designs (Epstein et al. 2003, Fairfield et al. 1984), Phase-Locked Loops in designs targeting FPGAs (Fischer and Drutarovsky 2012, Deak et al. 2015), Random access memories (Gyorfi et al. 2009) or multiple designs (Kohlbrenner and Gaj, 2004, Bucci et al., 2003, Golic, 2006, Tkacik, 2003) based on Ring Oscillators as a source of entropy.

As discussed in (Maes et al, 2012), PUF-based key generators try to fulfill two requirements on secure key generation and storage at once. The randomness of the PUF response comes from the manufacturing process variation, and it is intrinsically present in the device. There is no need for a protected non-volatile memory since the randomness is measured only when needed. However, the PUF output may slightly vary in different measurements, and it is still challenging to get static PUF output as required by cryptographic schemes. Existing PUF designs proposed for cryptographic applications include PUFKY based on a ring oscillator PUF (Maes et al, 2012) providing low-failure rate, generation of read-once keys (Kirkpatrick et al. 2010), single-chip secure processor for embedded systems (Suh et al., 2007), arbiter PUF for device authentication and secret key generation (Suh and Devadas, 2007) and others.

Our goal is to propose a secure communication and authentication method using a combined PUF/TRNG circuit that will allow secure generation of keys using both PUF and TRNG at the same time, maximizing benefits of each one. There have been several similar works published recently, however, the first attempts in using PUF for the device authentication were rather simple. In (Suh and Devadas, 2007) simple authentication against authentication authority was discussed, using pre-generated challenge-response pairs stored centrally. At the authentication time, the challenge is sent to the device and response then compared with the output. Same challenge cannot be reused again due to possible replay attacks.

More sophisticated PUF-based authentication protocols were reviewed in (Delvaux et al., 2014). The work of (Ozturk et al., 2008) using reprogrammable non-volatile memory; Hammouri et al. (Hammouri et al. , 2008) using two arbiter PUFs; protocol based on logically reconfigurable PUFs (Katzenbeisser et al. 2011) which allows to recycle the challenge tokens; Reverse Fuzzy Extractor (Van Herrewege et al., 2012) allowing mutual authentication; Slender PUF protocol (Majzoobi et al., 2012) that does not expose the full PUF responses, only the random subset instead; and Converse authentication Protocol (Kocabas et al., 2012) which provides one-way authentication of the server.

All of the protocols discussed above deal with authentication only, leaving the need for key establishment and secure communication open, which is also one of the main conclusions of the PUF authentication usage review (Delvaux et al., 2014), discussing the caveats of the PUF responses being not perfectly reproducible, small output space of strong PUFs or need of secure TRNG, that is substantial for most of the protocols.

Another concern is privacy of the authenticated devices. As discussed in (Bolotnyy and Robins, 2007), an algorithm is privacy preserving if an adversary cannot distinguish between any pair of devices, and thus, the PUF must be able to generate long chains of unique IDs (i.e., without repetitions). Possible approach to privacy preserving authentication protocol is presented in (Aysu et al., 2015), based on fuzzy extractor with helper-data construction technique based on TRNG.

In this paper we discuss protocols for authentication and secure communication utilizing PUF and TRNG. We show, it is advantageous to have single module that will allow generation of both TRNG and PUF at the same time, since it minimizes implementation requirements and operational resource consumption. The aim is not to require storing secrets on the IoT or embedded system itself to simplify key management on the simple hardware devices and microcontrollers.

3 PROPOSED APPROACH

When designing the embedded module for secure authentication and communication, the main goal is to simplify the key management on the endpoint embedded device itself. Thus, we propose to utilize the single circuit for key generation using PUF and TRNG as a basic building block of the module so that

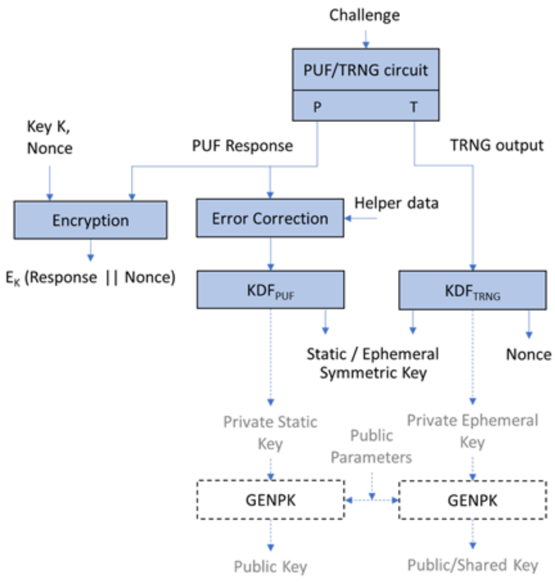


Figure 2: Embedded module for secure authentication and communication. KDF is a Key Derivation Function, GENPK generates a public key from a private key and public parameters. Error Correction is used to obtain a stable key material from the PUF Response (see Section 3.1). Functions covered in this paper are coloured in blue.

there is no need to store secrets on the hardware device.

The overall module depicted in Figure 2 provides PUF authentication and PUF/TRNG based key generation. For the authentication, the PUF is used, since it provides randomness intrinsically present in the device and utilizes the fact that the generated response is unique per device. Since there is a need for both static key, as well as ephemeral keys, combination of PUF and TRNG is used in this case – the PUF is used for generation of static (private) key that never leaves the device, thus utilizing all the advantages of PUF, while TRNG is used for generation of ephemeral, one-time keys, that are shared with other communicating parties.

Asymmetric schemes are suitable if the private key is easily generated from random sequence by a Key Derivation Function (KDF), such as PBKDF2 (RSA Laboratories, 2012). For example, ElGamal encryption (ElGamal, 1985) and DSA/ECDSA signature schemes can be used, if good quality public parameters are chosen (generation of the public key from a private key si denoted as GENPK in Figure 2). On the contrary, an RSA key requires more complex processing including secure prime generation. TRNG output is also used to generate random nonce and padding data. In this paper, we will focus only on symmetric schemes (with the exception of Algorithm 1).

The proposed protocols for authentication against a central authentication authority, and also mutual device-device authentication, are further discussed in detail in following sections.

3.1 Authentication against Central Authentication Authority

Before any communication is allowed the connected device must be properly authenticated. Since the PUF responses are unique per each device, and are intrinsically random, i makes PUF the ideal cryptographic primitive for device authentication. We propose simple and straightforward authentication protocol using pre-generated challenge-response pairs that can be easily implemented in hardware devices. This protocol does not require the PUF to have a large space of challenge-response pairs (it can be used even for one challenge-response pair). The authentication protocol consists of two phases – secure enrolment phase and authentication phase itself and is depicted in Algorithm 1.

The Enrolment phase is critical for the security of all protocols based on PUFs, and (analogous to biometric authentication methods) must be performed in a secure environment. We assume that a suitable environment can be created (for example, by separate physical access to the devices), but specific means are not elaborated in this paper.

During the Enrolment phase of Algorithm 1, the challenge/response pair(s) (C, R) are measured from the targeted device and securely stored at the central authenticating authority (AA), that can be either integrated into the gateway or be represented by separate device that the gateway is querying during authentication process. A database DB_{D_i} of the pairs (C, R) is created for each device D_i. Furthermore, the public key (PK_{AA}) of the authenticating authority is pre-set on the device, so as the authentication data can be securely transferred. For this purpose, an asymmetric scheme (ElGamal) can be used, as proposed in the section above. We assume that PK_{AA} is protected against unauthorized changes (by the tamper-evidence property of the PUF).

The first 4 steps of the Enrolment phase are common for all 3 algorithms presented in this paper. The database DB_{D_i} is used also in the Authentication phases of Algorithms 2 – 3.

```
Enrolment phase (Secure environment)
Common for Algorithms 1 – 3:
1. AA → D1: Challenges ($C_1$, $C_2$, ...)
2. D1:      $R_1$ = PUF($C_1$), $R_2$ = PUF($C_2$) ...
3. D1 → AA: Responses ($R_1$, $R_2$, ...)
4. AA:      Store ($C_i$, $R_i$) to $DB_{D1}$

Specific only for Algorithm 1:
5. AA → D1: Public key $PK_{AA}$
6. D1:      Store($PK_{AA}$)

Authentication phase for D1
1. AA:      Choose (C, R) from $DB_{D1}$
2. AA → D1: Challenge C, Nonce N
3. D1:      R' = PUF(C)
4. D1 → AA: CR = $E_{PK\_AA}$(R' || N)
5. AA:      (R', N') = $D_{SK\_AA}$(CR)
            Compare($R \cong R'$), Compare(N = N')
```

Algorithm 1: Enrolment and Authentication against central Authentication Authority.

The Authentication phase of Algorithm 1 is executed every time the device is connected to the network and needs to be authenticated. AA randomly chooses one of the challenges C and sends it together with the nonce value N to the device to be authenticated. The nonce value is used to prevent simple replay attacks and allows each challenge-response pair to be used repeatedly. On the device that is being authenticated the appropriate PUF response is generated, concatenated with nonce value and encrypted with public key of the Authenticating Authority. Authenticating Authority then compares (strictly) if the decrypted nonce value N' = N. Since the PUF response may slightly vary across various measurements, a predetermined number of faulty bits in R' is tolerated. If both match, the device is successfully authenticated. The disadvantage of Algorithm 1 is that it only performs authentication and does not provide a cryptographic context for future communication.

Authentication of a single device (D1) to the AA without asymmetric cryptography is depicted in Algorithm 2. (The Enrolment phase is the same as in Algorithm 1, steps 1. – 4.) This method includes generating a shared symmetric key K, which requires a stable error-free PUF output. This is achieved by using an error-correcting code (ECC), denoted in the algorithm by its functions Encode and Decode. This code must have enough redundancy and structure to correct the maximum amount of errors assumed in the PUF when operated under various conditions (voltage, temperature etc.).

Choosing a suitable ECC depends on the bit error rate and length of PUF response while meeting the required corrected output length. The computational power of the device is also a limiting factor. In the case of "lightweight" devices, simple codes (such as a repetition code) are preferable.

Authentication phase – using symmetric cipher

```
1.  D1 → AA: Call(D1)

2.  AA:   r = TRNG()
3.        Choose (C, R) from $DB_{D1}$
4.        H = R ⊕ Encode(r)
5.        K = KDF(r)

6.  AA → D1: Challenge C, Helper string H

7.  D1:   R' = PUF(C)
8.        r = Decode(R' ⊕ H)
9.        K = KDF(r)

10. D1 ↔ AA: Authentication + Encryption with K
```

Algorithm 2: Authentication of a device D1 to the AA.

The helper string H is a distance from the raw PUF response R to the random codeword Encode(r). It is computed by the AA (step 4 of Algorithm 2). The device then uses it to recover the key material (step 8), and subsequently derive the key K.

The shared key K can be used for authentication and encrypted communication, as opposed to Algorithm 1, which covers only authentication, limiting its usefulness. On the other hand, Algorithm 1 does not require the generation of a helper string, nor does it need any error correction codes.

3.2 Mutual Device Authentication

Not only the device needs to be authenticated to central authority when connected to the network, the devices must be mutually authenticated before they start to communicate, as well. Similarly, as in the previous case, central authenticating authority stores the pre-generated challenge-response pair(s), and acts as trusted 3rd party. This time though, a shared symmetric key is established between the two devices, and a conventional symmetric authenticated and encrypted session can follow afterwards. The goal is to use the PUFs in both devices D1 and D2, but not transmit any PUF response over the network. By using the one-wayness of the hash functions used, no device gets to know other device's PUF response, even if it monitors all communication. An error correcting code is used to ensure stable PUF outputs. The codewords are selected randomly from the code space by the AA. The overall process is described in Algorithm 3.

Mutual authentication of D1 and D2 using AA

```
1.  D1 → AA: Call(D1, D2)

2.  AA:   $r_{D1}$ = TRNG()
3.        $r_{D2}$ = TRNG()
4.        Choose ($C_{D1}$, $R_{D1}$) from $DB_{D1}$
5.        Choose ($C_{D2}$, $R_{D2}$) from $DB_{D2}$
6.        $H_{D1}$ = $R_{D1}$ ⊕ Encode($r_{D1}$)
7.        $H_{D2}$ = $R_{D2}$ ⊕ Encode($r_{D2}$)
8.        r = Hash($r_{D1}$) ⊕ Hash($r_{D2}$)

9.  AA → D1: ($C_{D1}$, $H_{D1}$, r)
10. AA → D2: Call(D1, D2) , ($C_{D2}$, $H_{D2}$, r)

11. D1:   $R'_{D1}$ = PUF($C_{D1}$)
12.       $r_{D1}$ = Decode($R'_{D1}$ ⊕ $H_{D1}$)
13.       Hash($r_{D2}$) = Hash($r_{D1}$) ⊕ r
14.       K = KDF(Hash($r_{D1}$) || Hash($r_{D2}$))

15. D2:   $R'_{D2}$ = PUF($C_{D2}$)
16.       $r_{D2}$ = Decode($R'_{D2}$ ⊕ $H_{D2}$)
17.       Hash($r_{D1}$) = Hash($r_{D2}$) ⊕ r
18.       K = KDF(Hash($r_{D1}$) || Hash($r_{D2}$))

19. D1 ↔ D2: Authentication + Encryption with K
```

Algorithm 3: Mutual device authentication and secure communication.

Let us assume that D1 wants to authenticate with D2 and set up a secure communication channel. D1 initiates the process by calling the AA with the identification of D1 and D2 (CALL(D1, D2)). AA contains the complete table of challenges and responses (C_{D1}, R_{D1} etc.). An error correcting code is chosen that can correct enough errors to make the PUF response stable, with the corresponding functions Encode and Decode. AA generates two random components r_{D1}, r_{D2} from the set of preimages, and encodes them, thereby forming randomly chosen codewords. The code length should correspond to the PUF response length. Helper strings H_{D1} and H_{D2} are created by XORing the expected PUF response (R_{D1}, R_{D2}) to the corresponding codeword. The two random components are hashed and the hashes XORed to form r.

To each of the devices, a triplet (C_{Di}, H_{Di}, r) with the challenge, helper string, and r is sent. Also, in step 10, AA relays the request for communication from D1 to D2. Each of the devices challenges its own PUF to get the response (R'_{D1}, R'_{D2}). By XORing the response with the corresponding helper string (H_{D1}, H_{D2}), resulting with a codeword with errors, which is then corrected by the Decode function. This way, each device recovers its component (r_{D1}, r_{D2}). D1 recovers the value Hash(r_{D2}) by XORing r with the hash of its r_{D1}, and vice versa. Moreover, both devices know the hashes of r_{D1} and r_{D2}, and can derive the shared key K by applying a key derivation function KDF on the concatenation of the hashes.

The hashing of r_{D1}, r_{D2} is done to hide the PUF responses from the other device. If D1 monitors the communication, it will know (C_{D1}, C_{D2}, H_{D1}, H_{D2}, r). It can recover r_{D1}, and if the hashing were not done, and r would be equal to $r_{D1} \oplus r_{D2}$ directly, D1 would compute r_{D2}, and using the helper string H_{D2}, it could discover the PUF response R_{D2}. We would have to either trust all devices in the network or use all challenges only once and discard them. In our case, because we do use hashing of r_{D1}, r_{D2}, D1 only gets Hash(r_{D2}), and the one-wayness of the hash function prevents it from discovering R_{D2}. Thus, we can reuse the challenges for future authentications.

PUF response correction code choice depends on the number of bit flips inherent in the PUF operation. The code length and codeword distance determine the number of information bits, thus the length of r_{D1}, r_{D2}, and limit the entropy contained in r. By using the same challenge with multiple random r_{Di}, we can extract more bits of entropy from the PUF. The entropy of the resulting shared key K is determined by the properties of used hash functions and KDF, and the inputs. If chosen correctly, it is as high as the entropies of r_{D1}, r_{D2}. The key K is always derived from randomly chosen codewords, and therefore for the same PUF challenges (C_{D1}, C_{D2}), a different K is obtained.

3.3 Secure Communication

After the authentication process described in the previous section, a shared key is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using block ciphers such as AES. Several lightweight block ciphers suitable for embedded systems or sensor networks has been proposed, such as PRESENT (Bogdanov et al., 2007, McKay, 2017) with an 80-bit key. This allows generating the key in a single run of PUF circuit for most of the PUF designs and implementations, with no further stretching needed.

All presented algorithms in this Section utilized only PUF on the side of the devices and TRNG was used on AA. TRNG functionality on the devices is used after the secure channel establishment (steps 10 and 19) in dependence on the communication protocols. Random numbers are needed in many classical authentication protocols (Menezes et al., 1996, chapter 12), as well as modern internet standards such as DTLS (Tschofenig et al., 2016).

4 CASE STUDY

As arises from previous section, both TRNGs and PUFs have different characteristics that are advantageous in different applications. Thus, various implementations of cryptographic systems can take an advantage from a universal circuit for generation of PUF and TRNG at the same time, that allows secure generation of symmetric (session) keys (and potentially also asymmetric (private) keys). Such

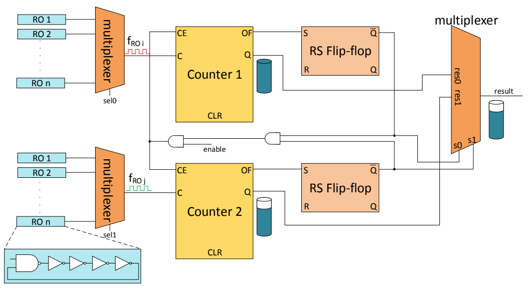


Figure 3: PUF/TRNG circuit based on Ring Oscillators, serving as basic building block for proposed authentication and secure communication scheme (Kodýtek et al., 2015, Buchovecká et al., 2017).

PUF/TRNG based on Ring Oscillators – ROPUF circuit was presented in our previous work (Kodýtek et al., 2015, Kodýtek et al., 2016, Buchovecká et al. 2016, Buchovecká et al., 2017), so the idea of the single RO circuit can be used both for PUF and TRNG generation was validated. This circuit is depicted in Figure 3.

In order to validate the proposed authentication process outlined in Section 3, we performed an experiment on one device containing the ROPUF design (Kodýtek et al., 2015, Buchovecká et al., 2017). For this purpose, we used a ROPUF design that consisted of 2 groups of ring oscillators (ROs), each group contained 150 ROs. Only ROs from different groups were selected to form a pair, which was then used to generate part of the PUF response. We extracted 3 bits from each RO pair and enhanced the stability of the PUF output by applying Gray code on these bits (Kodýtek et al., 2016). Finally, to create the PUF response, the selected bits from all of the RO pairs are concatenated.

In the first case, we generated the PUF responses from 150 pairs of ROs (each RO from each group was used only once), in the other, each RO was used five times (one RO from the first group is paired with 5 ROs from the other group) resulting in 750 RO pairs. These two setups achieved 450 and 2250 bits of PUF response respectively. In both cases, we performed 1000 measurements, from which we obtained a majority PUF response - R_{Di} (we determined the majority for each position of the PUF output).

In our experiment, the block length of 9 bits proved to be sufficient for the repetition code. In order to create the helper string $H_{Di,}$ we need to generate 50 or 250 random bits (r_{Di}) that are then encoded by the repetition code and XORed with the major PUF output, forming the helper string H_{Di}. This process is related to steps 2 and 4 in Algorithm 2.

The example using a simple repetition code with 5-bit block length is depicted on Figure 4. On the device, the PUF generates a response R'_{Di} that is corrected by the helper string H_{Di}, corresponding to steps 7 and 8 in Algorithm 2. After correction, we obtained 50 and 250 bits respectively. These bits can be used to create a cryptographic key. For Algorithm 2, we can simply represent KDF as the selection of the first 128 bits (from r_{Di}) for symmetric cipher AES.

The same can be applied for Algorithm 3, where two devices are authenticating each other. However, this algorithm is more complex, since it requires implementation of suitable hash function. In case of Algorithm 1, no KDF is needed, since the AA's public key is stored on the device and PUF is not used to derive any cryptographic key.

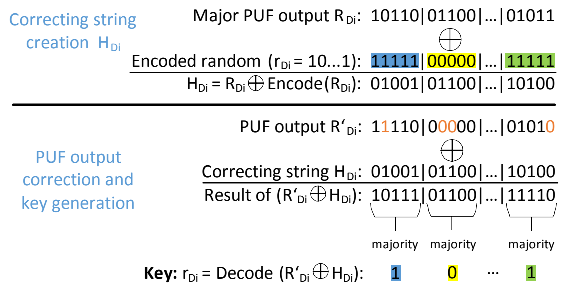

Correcting string creation H_{Di}
Major PUF output R_{Di}: 10110|01100|...|01011
$\oplus$
Encoded random (r_{Di} = 10...1): 11111|00000|...|11111
$H_{Di} = R_{Di} \oplus$ Encode(R_{Di}): 01001|01100|...|10100
PUF output correction and key generation
PUF output R'_{Di}: 11110|00000|...|01010
$\oplus$
Correcting string H_{Di}: 01001|01100|...|10100
Result of ($R'_{Di} \oplus H_{Di}$): 10111|01100|...|11110
majority
majority
majority
Key: r_{Di} = Decode ($R'_{Di} \oplus H_{Di}$): 1 0 ... 1

Figure 4: Example of a simple repetition code with 5-bit groups.

To increase the number of bits after correction, we can either use a more efficient error correcting code or we can reuse the same challenge multiple times with a new random codeword each time. The experiment showed and confirmed that it is possible to generate key material for the proposed protocols, using the state of the art PUF/TRNG designs, in sufficient length and quality.

5 CONCLUSIONS

In the paper, we discussed the need for the proper key management of cryptographic keys on the embedded devices and further proposed the design of the module for secure authentication and communication that fulfills the requirements for the secure generation and storage of the cryptographic keys, including proposal of basic authentication and secure communication protocols.

For the authentication, several protocols based on pre-generated PUF challenge/response values are proposed. Since the PUF responses are unique per each device and are intrinsically random, PUF is ideal cryptographic primitive for this purpose. Three variants of the protocol are discussed – authentication against central authority using PUF challenge and encrypted response, and two variants of authentication that use the PUF for key generation – single device authentication, and mutual device authentication.

After the authentication process, a shared key between the devices is established. At this point, a conventional symmetric authentication and session key derivation process can be performed using conventional or lightweight block ciphers as needed.

Further, we discussed the case study and suggested possible implementation of the module for secure communication and authentication, using

ROPUF/TRNG circuit. As it was presented and validated in previous work (Kodýtek et al., 2015, Kodýtek et al., 2016, Buchovecká et al., 2016, Buchovecká et al., 2017) a pair of RO circuits can be used both for PUF and TRNG generation, thus serve as basic building block for the module. Moreover, it is possible to generate the sequence long enough for the key generation in one run of ROPUF/TRNG circuit.

In the paper we have shown the possibilities of securing communication and authentication of the embedded systems, using PUF and TRNG for secure key generation, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and microcontrollers.

Future work will be devoted to secure communication with a suitable asymmetric encryption scheme, using both PUF for generation of private key, as well as TRNG for generation of ephemeral keys, making use of the randomness already intrinsically present in the device. Thanks to PUF, the private key is generated when needed, thus there is no need for storing secrets on the device itself.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/ 0000765 "Research Center for Informatics".

REFERENCES

Aysu, A., Gulcan, E., Moriyama, D., Schaumont, P., & Yung, M. (2015). End-to-end design of a PUF-based privacy preserving authentication protocol. *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., & Vikkelsoe, C. (2007). PRESENT: An ultra-lightweight block cipher. *International workshop on cryptographic hardware and embedded systems* (pp. 450-466). Springer, Berlin, Heidelberg.

Bolotnyy L., Robins G. (2007). Physically unclonable function-based security and privacy in RFID systems. *Pervasive Computing and Communications, 2007. PerCom'07*. Fifth Annual IEEE International Conference on. IEEE.

Bucci, M., Germani, L., Luzzi, R., Trifiletti, A., & Varanonuovo, M. (2003). A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC. *Computers, IEEE Transactions* on 52.4 (2003): 403-409.

Buchovecká, S., Kodýtek, F., Lórencz, R., Buček J. (2016) True Random Number Generator Based on ROPUF Circuit. *Digital System Design (DSD), 2016 Euromicro Conference on*. IEEE.

Buchovecká, S., Kodýtek, F., Lórencz, R., Buček J. (2017). True random number generator based on ring oscillator PUF circuit. *Microprocessors and Microsystems* 53 (2017): 33-41.

Chan H., Gligor V. D., Perrig A., Muralidharan G. (2005). On the distribution and revocation of cryptographic keys in sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):233–247.

Deak N., Gyorfi T., Marton K., Vacariu L., Cret O. (2015). Highly efficcient true random number generator in FPGA devices using phase-locked loops. *20th International Conference on Control Systems and Computer Science*, pages 453–458. IEEE.

Delvaux, J., Gu, D., Schellekens, D., & Verbauwhede, I. (2014). Secure lightweight entity authentication with strong PUFs: Mission impossible? In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg. p. 451-475.

ElGamal T. (1985). A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472.

Epstein M., Hars L., Krasinski R., Rosner M., and Zheng H. (2003). Design and implementation of a true random number generator based on digital circuit artifacts. *In International Workshop on Cryptographic Hardware and Embedded Systems*, pages 152–165. Springer.

Fairfield R. C., Mortenson R. L., and Coulthart K. B. (1984). An LSI random number generator (rng). *Workshop on the Theory and Application of Cryptographic Techniques*, pages 203–230. Springer.

Fischer V. (2012). A closer look at security in random number generators design. *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 167–182. Springer.

Fischer V., Drutarovský M. (2002). True random number generator embedded in reconfigurable hardware. *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 415–430. Springer.

Golic J. D. J. (2006), New Methods for Digital Generation and Postprocessing of Random Data, *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217-1229.

Gyorfi T., Cret O., and Suciu A. (2009). High performance true random number generator based on fpga block rams. *Parallel & Distributed Processing, 2009.* IPDPS 2009. IEEE International Symposium on, pages 1–8. IEEE.

Hammouri, G., Öztürk, E., Sunar, B. (2008). A tamper-proof and lightweight authentication scheme. *Journal Pervasive and Mobile Computing* 6(4).

Handschuh H., Schrijen G.-J., and Tuyls P. (2010). Hardware intrinsic security from physically unclonable functions. *Towards Hardware-Intrinsic Security*, pages 39–53. Springer.

Haroon, A., Akram, S., Shah, M. A., & Wahid, A. (2017). E-lithe: A lightweight secure dtls for iot. In 2017 *IEEE 86th Vehicular Technology Conference (VTC-Fall)* (pp. 1-5). IEEE.

Katzenbeisser, S., Kocabaş, Ü., Van Der Leest, V., Sadeghi, A. R., Schrijen, G. J., & Wachsmann, C. (2011). Recyclable PUFs: Logically reconfigurable PUFs. *Journal of Cryptographic Engineering*, 1(3), pp. 177–186.

Kirkpatrick M. S., Bertino E., and Kerr S. (2010). PUF ROKs: generating read-once keys from physically unclonable functions. *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*. ACM.

Kocabaş, Ü., Peter, A., Katzenbeisser, S., Sadeghi, A.-R. (2012). Converse PUF-Based Authentication. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) *Trust 2012. LNCS*, vol. 7344, pp. 142–158. Springer, Heidelberg.

Kodýtek, F., Lórencz, R. (2015). A design of ring oscillator based PUF on FPGA. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2015 IEEE 18th International Symposium on. IEEE.

Kodýtek, F., Lórencz, R., Buček J. (2016) Improved ring oscillator PUF on FPGA and its properties. *Microprocessors and Microsystems*.

Kohlbrenner P., Gaj K. (2004). An embedded true random number generator for FPGAs. *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. ACM.

McKay K. A. (2017). Report on Lightweight Cryptography – NIST publication, available online https://doi.org/10.6028/NIST.IR.8114

Maes R., Van Herrewege A., and Verbauwhede I. (2012). PUFKY: a fully functional PUF-based cryptographic key generator. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 302–319. Springer.

Majzoobi, M., Rostami, M., Koushanfar, F., Wallach, D.S., Devadas, S. (2012). Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching. In: *IEEE Symposium on Security and Privacy* (SP), pp. 33–44.

Malina L., Hajny J., Fujdiak R., and Hosek J. (2016). On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks*, 102:83– 95.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC press.

Öztürk, E., Hammouri, G., Sunar, B. (2008). Towards Robust Low-Cost Authentication for Pervasive Devices. In: *IEEE Conference on Pervasive Computing and Communications*, PerCom.

Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. (2002). Physical one-way functions. *Science*, 297(5589):2026–2030.

Raza, S., Shafagh, H., Hewage, K., Hummen, R., & Voigt, T. (2013). Lithe: Lightweight secure CoAP for the internet of things. *IEEE Sensors Journal*, 13(10), 3711-3720.

Roman R., Zhou J., and Lopez J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279.

RSA Laboratories: PKCS #5 V2.1: Password Based Cryptography Standard (2012)

Schindler W. (2009). Random number generators for cryptographic applications. In *Cryptographic Engineering*, pages 5–23. Springer.

Schleiffer Ch., Wolf M., Weimerskirch A., and Wolleschensky L. (2013). Secure key management-a key feature for modern vehicle electronics. Technical report, SAE Technical Paper.

Sicari S., Rizzardi A., Grieco L. A., Coen-Porisini A. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164.

Sklavos, N, Zaharakis I. D. (2016). Cryptography and Security in Internet of Things (IoTs): Models, Schemes, and Implementations, In *IEEE proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security* (NTMS'16), Larnaca, Cyprus

Suh E. G., Devadas S. (2007). Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM.

Suh E. G., O'Donnell Ch. W., and Devadas S. (2007). Aegis: A single-chip secure processor. *IEEE Design & Test of Computers* 24.6.

Tkacik T. E. (2003), A hardware random number generator, in Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '02), B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, pp. 450–453, Springer, Redwood Shores, Calif, USA.

Tschofenig, H. and T. Fossati," Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things. RFC 7925, July 2016.

Van Herrewege, A., Katzenbeisser, S., Maes, R., Peeters, R., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C. (2012). Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs. In: Keromytis, A.D. (ed.) *FC 2012. LNCS*, vol. 7397, pp. 374–389. Springer, Heidelberg.

CHAPTER 5

Conclusions

5.1 Summary

For the security of modern cryptographic systems it is inevitable to protect the confidential keys used for the cryptographic protocols, as well as to guarantee the required qualities such as randomness and unpredictability. Implementation of proper methods for cryptographic key generation and their secure storage in embedded devices (including programmable logic devices) is of significant importance. Moreover, it is necessary to use the cryptography and corresponding keys in proper manner. Since one of the main requirement on the cryptographic keys is their unpredictability and randomness, thus the Random Number Generators (RNGs) are often utilized for the key generation, however, numerous works dealing with Physical Unclonable Functions (PUFs) for key generation had been published recently, proposing PUFs as another possible approach for key generation. Moreover, PUFs usage is promising to solve the issue of secure storage of cryptographic keys – Instead of storing the key in memory, the key is generated at the time it is needed.

In **RP2** we dealt with possibilities of attack on TRNG and performed both invasive and non-invasive attacks on the Atmega-based TRNG design, with the idea to basically force the generator to generate less unique values and thus lower the entropy. We explored both possibilities of both non-invasive and invasive attack successfully. The lesson learned is, that it is necessary to test the TRNG not only in design phase, but also in operations to ensure the quality of generated key material. The **RP1** was dedicated to the testing of TRNG, using not only general test suites, but also looking at characteristics arising from the design itself.

In **RP3** and **RP4** we proposed the the TRNG design based on ROPUF, to prove that it is possible to create a universal circuit for generating PUF and TRNG at the same time, for further cryptographic applications. Both PUF and TRNG utilizes Ring Oscillators (RO) as an entropy source, since it is advantageous for proper implementation of various cryptographic protocols to have the ability to generate both random number (e.g. for symmetric keys), as well as PUF (for secure generation and storage of asymmetric keys) at the same time.

We further used proposed PUF/TRNG combined circuit in **RP5** to design the embedded module for secure authentication and communication, including the authentication and communication algorithms, with the main goal is to simplify the key management on the endpoint embedded device itself. Implementation of proper methods for authentication and secure communication, including secure management of key material in lightweight devices, is of significant importance, especially with rise of IoT devices. The use of cryptography and corresponding keys in proper manner is one of the leading problems, when talking about devices with limited computing. The single circuit was utilized for key generation using PUF and TRNG as a basic building block of the module so that there is no need to store secrets on the hardware device.

5.2 Contributions of the Dissertation Thesis

In particular, the main contributions of the dissertation thesis are as follows:

1. Discussion over TRNG testing and its practical applications in the hardware random number generators testing. With practical attack on the RNG we showed the importance of the online testing of generated sequence as well as focus on the characteristics arising from the design itself,
2. Proposal of TRNG design sbased on ROPUF circuit, enabling the simultaneous generation of PUF and TRNG using the same hardware component, suitable even for simple micro-controllers and embedded devices,
3. Proposal of the protocols for lightweight authentication and secure communication for IoT and embedded devices showing the possibilities of securing communication and authentication of the embedded systems, using PUF/TRNG combined circuit as a basic building block, without requirement to store secrets on the device itself, thus allowing to significantly simplify the problem of key management on the simple hardware devices and micro-controllers.

5.3 Future Work

The author of the dissertation thesis suggests to explore the following:

- In further work, more attention needs to be paid to systematic approach on overall key management, key generation, storage and usage on hardware devices, embedded systems and interconnected devices within Internet of Things – the requirements should be formally defined.
- Based on the formally defined requirements for key management, cost-effective and lightweight Hadrware Security Module (HSM) design, that can be even built-in into the embedded systems and simple hardware devices, can be developed. Using hardware

cryptographic primitives such as TRNGs and PUFs, however, allows to maintain high level of security.

Bibliography

[1] Apple. *iOS Security*, 2018.

[2] Avaroğlu, Erdinç, Tuncer, Taner, Özer, A Bedri, Ergen, Burhan, and Türk, Mustafa. *A novel chaos-based post-processing for TRNG.* Nonlinear Dynamics, 81(1-2):189–199, 2015.

[3] Aysu, Aydin, Gulcan, Ege, Moriyama, Daisuke, Schaumont, Patrick, and Yung, Moti. *End-to-end design of a PUF-based privacy preserving authentication protocol.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 556–576. Springer, 2015.

[4] Barak, Boaz, Shaltiel, Ronen, and Tromer, Eran. *True random number generators secure in a changing environment.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 166–180. Springer, 2003.

[5] Barker, Elaine and Roginsky, Allen. *Recommendation for cryptographic key generation.* NIST Special Publication, 800:133, 2012.

[6] Bassham, Lawrence E, Rukhin, Andrew L, Soto, Juan, Nechvatal, James R, Smid, Miles E, Leigh, Stefan D, Levenson, M, Vangel, M, Heckert, Nathanael A, and Banks, D L. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications— NIST.* Tech. rep., 2010.

[7] Baudet, Mathieu, Lubicz, David, Micolod, Julien, and Tassiaux, André. *On the security of oscillator-based random number generators.* Journal of cryptology, 24(2):398–425, 2011.

[8] Bellido, MJ, Acosta, AJ, Valencia, M, Barriga, A, and Huertas, JL. *Simple binary random number generator.* Electronics Letters, 28(7):617–618, 1992.

[9] Blaze, Matt. *A cryptographic file system for UNIX.* In Proceedings of the 1st ACM conference on Computer and communications security, pp. 9–16. ACM, 1993.

[10] Bogdanov, Andrey, Knudsen, Lars R, Leander, Gregor, Paar, Christof, Poschmann, Axel, Robshaw, Matthew JB, Seurin, Yannick, and Vikkelsoe, Charlotte. *PRESENT: An ultra-lightweight block cipher.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 450–466. Springer, 2007.

[11] Bond, Mike, Choudary, Omar, Murdoch, Steven J, Skorobogatov, Sergei, and Anderson, Ross. *Chip and Skim: cloning EMV cards with the pre-play attack.* In 2014 IEEE Symposium on Security and Privacy, pp. 49–64. IEEE, 2014.

[12] Bösch, Christoph, Guajardo, Jorge, Sadeghi, Ahmad-Reza, Shokrollahi, Jamshid, and Tuyls, Pim. *Efficient helper data key extractor on FPGAs.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 181–197. Springer, 2008.

[13] Bossuet, Lilian, Ngo, Xuan Thuy, Cherif, Zouha, and Fischer, Viktor. *A PUF based on a transient effect ring oscillator and insensitive to locking phenomenon.* IEEE Transactions on Emerging Topics in Computing, 2(1):30–36, 2013.

[14] Bucci, Marco, Germani, Lucia, Luzzi, Raimondo, Trifiletti, Alessandro, and Varanonuovo, Mario. *A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC.* IEEE transactions on computers, 52(4):403–409, 2003.

[15] Caddy, Tom. *FIPS 140-2.*, 2005.

[16] Chan, Haowen, Gligor, Virgil D, Perrig, Adrian, and Muralidharan, Gautam. *On the distribution and revocation of cryptographic keys in sensor networks.* IEEE Transactions on Dependable and Secure Computing, 2(3):233–247, 2005.

[17] Chen, Qingqing, Csaba, György, Ju, Xueming, Natarajan, Srinivas Bangalore, Lugli, Paolo, Stutzmann, Martin, Schlichtmann, Ulf, and Rührmair, Ulrich. *Analog circuits for physical cryptography.* In Proceedings of the 2009 12th International symposium on integrated circuits, pp. 121–124. IEEE, 2009.

[18] Consult, SEC. *House of Keys: Industry-Wide HTTPS Certificate and SSH Key Reuse Endangers Millions of Devices Worldwide.* Blog, Nov, 25, 2015.

[19] Davies, Robert. *Hardware random number generators.* 2000.

[20] Delvaux, Jeroen, Gu, Dawu, Schellekens, Dries, and Verbauwhede, Ingrid. *Secure lightweight entity authentication with strong PUFs: Mission impossible?* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 451–475. Springer, 2014.

[21] Dichtl, Markus. *Bad and good ways of post-processing biased physical random numbers.* In International Workshop on Fast Software Encryption, pp. 137–152. Springer, 2007.

[22] Dichtl, Markus. *Fibonacci Ring Oscillators as True Random Number Generators-A Security Risk.* IACR Cryptology ePrint Archive, 2015:270, 2015.

[23] Dichtl, Markus and Golić, Jovan Dj. *High-speed true random number generation with logic gates only.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 45–62. Springer, 2007.

[24] Diffie, Whitfield and Hellman, Martin. *New directions in cryptography.* IEEE transactions on Information Theory, 22(6):644–654, 1976.

[25] Dodis, Yevgeniy and Oliveira, Roberto. *On extracting private randomness over a public channel.* In Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques, pp. 252–263. Springer, 2003.

[26] Dodis, Yevgeniy, Ostrovsky, Rafail, Reyzin, Leonid, and Smith, Adam. *Fuzzy extractors: How to generate strong keys from biometrics and other noisy data.* SIAM journal on computing, 38(1):97–139, 2008.

[27] Elenkov, Nikolay. Android security internals: An in-depth guide to Android's security architecture. No Starch Press, 2014.

[28] ElGamal, Taher. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE transactions on information theory, 31(4):469–472, 1985.

[29] Epstein, Michael, Hars, Laszlo, Krasinski, Raymond, Rosner, Martin, and Zheng, Hao. *Design and implementation of a true random number generator based on digital circuit artifacts.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 152–165. Springer, 2003.

[30] FIPS. *186-2. digital signature standard (DSS).* National Institute of Standards and Technology (NIST), 2000.

[31] Fischer, Viktor. *A closer look at security in random number generators design.* In International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 167–182. Springer, 2012.

[32] Fischer, Viktor and Lubicz, David. *Embedded evaluation of randomness in oscillator based elementary TRNG.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 527–543. Springer, 2014.

[33] Gassend, Blaise, Clarke, Dwaine, Van Dijk, Marten, and Devadas, Srinivas. *Silicon physical random functions.* In Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160. ACM, 2002.

[34] Gassend, Blaise, Lim, Daihyun, Clarke, Dwaine, Van Dijk, Marten, and Devadas, Srinivas. *Identification and authentication of integrated circuits.* Concurrency and Computation: Practice and Experience, 16(11):1077–1098, 2004.

[35] Golic, Jovan Dj. *New methods for digital generation and postprocessing of random data.* IEEE Transactions on Computers, 55(10):1217–1229, 2006.

[36] Guajardo, Jorge, Kumar, Sandeep S, Schrijen, Geert-Jan, and Tuyls, Pim. *FPGA intrinsic PUFs and their use for IP protection.* In International workshop on cryptographic hardware and embedded systems, pp. 63–80. Springer, 2007.

[37] Hammouri, Ghaith, Öztürk, Erdinç, and Sunar, Berk. *A tamper-proof and lightweight authentication scheme.* Pervasive and mobile computing, 4(6):807–818, 2008.

[38] Handschuh, Helena, Schrijen, Geert-Jan, and Tuyls, Pim. *Hardware intrinsic security from physically unclonable functions.* In Towards Hardware-Intrinsic Security, pp. 39–53. Springer, 2010.

[39] Hartley, Ralph VL. *Transmission of information.* The Bell System Technical Journal, 7(3):535–563, 1928.

[40] Hlavá, Josef, Lórencz, Róbert, and Hadácek, Martin. *True random number generation on an Atmel AVR microcontroller.* In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, vol. 2, pp. V2–493. IEEE, 2010.

[41] Holcomb, Daniel E, Burleson, Wayne P, Fu, Kevin, et al. *Initial SRAM state as a fingerprint and source of true random numbers for RFID tags.* In Proceedings of the Conference on RFID Security, vol. 7, p. 01. 2007.

[42] Hotz, George. *Console hacking 2010-ps3 epic fail.* In 27th Chaos Communications Congress. 2010.

[43] Jonsson, Jakob and Kaliski, Burt. *Public-key cryptography standards (PKCS)# 1: RSA cryptography specifications version 2.1.* 2003.

[44] Jun, Benjamin and Kocher, Paul. *The Intel random number generator.* Cryptography Research Inc. white paper, 27:1–8, 1999.

[45] Kaliski, Burt. *PKCS# 5: Password-based cryptography specification version 2.0.* 2000.

[46] Kalyanaraman, Mukund and Orshansky, Michael. *Novel strong PUF based on nonlinearity of MOSFET subthreshold operation.* In 2013 IEEE international symposium on hardware-oriented security and trust (HOST), pp. 13–18. IEEE, 2013.

[47] Kang, Hyunho, Hori, Yohei, Katashita, Toshihiro, Hagiwara, Manabu, and Iwamura, Keiichi. *Cryptographie key generation from PUF data using efficient fuzzy extractors.* In 16th International Conference on Advanced Communication Technology, pp. 23–26. IEEE, 2014.

[48] KANUPARTHI, Arun, KARRI, Ramesh, and ADDEPALLI, Sateesh. *Hardware and embedded security in the context of internet of things.* In Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles, pp. 61–64. ACM, 2013.

[49] KATZENBEISSER, Stefan, KOCABAŞ, Ünal, ROŽIĆ, Vladimir, SADEGHI, Ahmad-Reza, VERBAUWHEDE, Ingrid, and WACHSMANN, Christian. *PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 283–301. Springer, 2012.

[50] KATZENBEISSER, Stefan, KOCABAŞ, Ünal, VAN DER LEEST, Vincent, SADEGHI, Ahmad-Reza, SCHRIJEN, Geert-Jan, and WACHSMANN, Christian. *Recyclable pufs: Logically reconfigurable pufs.* Journal of Cryptographic Engineering, 1(3):177, 2011.

[51] KERCKHOFFS, Auguste. *La cryptographie militaire.* Journal des sciences militaires, 9:538, 1883.

[52] KILLMANN, Wolfgang and SCHINDLER, Werner. *A proposal for: Functionality classes for random number generators.* ser. BDI, Bonn, 2011.

[53] KNUTH, Donald E. Art of computer programming, volume 2: Seminumerical algorithms. Addison-Wesley Professional, 2014.

[54] KOCABAŞ, Ünal, PETER, Andreas, KATZENBEISSER, Stefan, and SADEGHI, Ahmad-Reza. *Converse PUF-based authentication.* In International Conference on Trust and Trustworthy Computing, pp. 142–158. Springer, 2012.

[55] KOCHER, Paul, LEE, Ruby, MCGRAW, Gary, RAGHUNATHAN, Anand, and MODERATOR-RAVI, Srivaths. *Security as a new dimension in embedded system design.* In Proceedings of the 41st annual Design Automation Conference, pp. 753–760. ACM, 2004.

[56] KODỲTEK, Filip and LÓRENCZ, Róbert. *A design of ring oscillator based PUF on FPGA.* In 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pp. 37–42. IEEE, 2015.

[57] KODỲTEK, Filip, LÓRENCZ, Róbert, and BUČEK, Jiří. *Improved ring oscillator PUF on FPGA and its properties.* Microprocessors and Microsystems, 47:55–63, 2016.

[58] KOEBERL, Patrick, LI, Jiangtao, RAJAN, Anand, and WU, Wei. *Entropy loss in PUF-based key generation schemes: The repetition code pitfall.* In 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 44–49. IEEE, 2014.

[59] Kumar, Sandeep S, Guajardo, Jorge, Maes, Roel, Schrijen, Geert-Jan, and Tuyls, Pim. *The butterfly PUF protecting IP on every FPGA.* In 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 67–70. IEEE, 2008.

[60] Lofstrom, Keith, Daasch, W Robert, and Taylor, Donald. *IC identification circuit using device mismatch.* In 2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056), pp. 372–373. IEEE, 2000.

[61] Ma, Yuan, Lin, Jingqiang, Chen, Tianyu, Xu, Changwei, Liu, Zongbin, and Jing, Jiwu. *Entropy evaluation for oscillator-based true random number generators.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 544–561. Springer, 2014.

[62] Maes, Roel. *Physically Unclonable Functions: Properties.* In Physically Unclonable Functions, pp. 49–80. Springer, 2013.

[63] Maes, Roel. Physically unclonable functions. Springer, 2016.

[64] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. *Intrinsic PUFs from flip-flops on reconfigurable devices.* In 3rd Benelux workshop on information and system security (WISSec 2008), vol. 17, p. 2008. 2008.

[65] Maes, Roel, Van Herrewege, Anthony, and Verbauwhede, Ingrid. *PUFKY: A fully functional PUF-based cryptographic key generator.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 302–319. Springer, 2012.

[66] Maiti, Abhranil and Schaumont, Patrick. *Improving the quality of a physical unclonable function using configurable ring oscillators.* In 2009 International Conference on Field Programmable Logic and Applications, pp. 703–707. IEEE, 2009.

[67] Maiti, Abhranil and Schaumont, Patrick. *Improved ring oscillator PUF: An FPGA-friendly secure primitive.* Journal of cryptology, 24(2):375–397, 2011.

[68] Majzoobi, Mehrdad, Ghiaasi, Golsa, Koushanfar, Farinaz, and Nassif, Sani R. *Ultra-low power current-based PUF.* In 2011 IEEE international symposium of circuits and systems (ISCAS), pp. 2071–2074. IEEE, 2011.

[69] Majzoobi, Mehrdad, Koushanfar, Farinaz, and Potkonjak, Miodrag. *Testing techniques for hardware security.* In 2008 IEEE International Test Conference, pp. 1–10. IEEE, 2008.

[70] Majzoobi, Mehrdad, Rostami, Masoud, Koushanfar, Farinaz, Wallach, Dan S, and Devadas, Srinivas. *Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching.* In 2012 IEEE Symposium on Security and Privacy Workshops, pp. 33–44. IEEE, 2012.

[71] Malina, Lukas, Hajny, Jan, Fujdiak, Radek, and Hosek, Jiri. *On perspective of security and privacy-preserving solutions in the internet of things.* Computer Networks, 102:83–95, 2016.

[72] Marsaglia, George. *The Marsaglia random number CDROM with the DIEHARD battery of tests of randomness.* Distributed by the author (geo@ stat. fsu. edu) from Florida State University, 1996.

[73] Mavrovouniotis, Stathis and Ganley, Mick. *Hardware security modules.* In Secure Smart Embedded Devices, Platforms and Applications, pp. 383–405. Springer, 2014.

[74] Merli, Dominik, Stumpf, Frederic, and Sigl, Georg. *Protecting PUF Error Correction by Codeword Masking.* IACR Cryptology ePrint Archive, 2013:334, 2013.

[75] Microsoft. *Key Storage and Retrieval*, 2018.

[76] Mispan, Mohd Syafiq, Halak, Basel, Chen, Zufu, and Zwolinski, Mark. *TCO-PUF: A subthreshold physical unclonable function.* In 2015 11th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME), pp. 105–108. IEEE, 2015.

[77] Moriyama, Daisuke, Matsuo, Shin'ichiro, and Yung, Moti. *PUF-based RFID authentication secure and private under memory leakage.* IACR Cryptol. ePrint Arch, 3:61–83, 2013.

[78] Morozov, Sergey, Maiti, Abhranil, and Schaumont, Patrick. *An analysis of delay based PUF implementations on FPGA.* In International Symposium on Applied Reconfigurable Computing, pp. 382–387. Springer, 2010.

[79] Newell, G. Richard. *Measurement of FPGA ring oscillator noise, and analysis using the Allan Variance method*, 2011. Unpublished Lecture.

[80] Nohl, Karsten, Evans, David, Starbug, Starbug, and Plötz, Henryk. *Reverse-Engineering a Cryptographic RFID Tag.* In USENIX security symposium, vol. 28. 2008.

[81] Nyquist, Harry. *Certain topics in telegraph transmission theory.* Transactions of the American Institute of Electrical Engineers, 47(2):617–644, 1928.

[82] Öztürk, Erdinç, Hammouri, Ghaith, and Sunar, Berk. *Towards robust low cost authentication for pervasive devices.* In 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 170–178. IEEE, 2008.

[83] Paillier, Pascal. *Public-key cryptosystems based on composite degree residuosity classes.* In International Conference on the Theory and Applications of Cryptographic Techniques, pp. 223–238. Springer, 1999.

[84] Pappu, Ravikanth, Recht, Ben, Taylor, Jason, and Gershenfeld, Neil. *Physical one-way functions.* Science, 297(5589):2026–2030, 2002.

[85] Park, Hojoong, Yeom, Yongjin, and Kang, Ju-Sung. *A Lightweight BCH Code Corrector of TRNG with Measurable Dependence.* Security and Communication Networks, 2019, 2019.

[86] Perić, Miroslav, Milićević, Predrag, Banjac, Zoran, Orlić, Vladimir, and Milićević, Saša. *High speed random number generator for section key generation in encryption devices.* In 2013 21st Telecommunications Forum Telfor (TELFOR), pp. 117–120. IEEE, 2013.

[87] Puntin, Daniele, Stanzione, Stefano, and Iannaccone, Giuseppe. *CMOS unclonable system for secure authentication based on device variability.* In ESSCIRC 2008-34th European Solid-State Circuits Conference, pp. 130–133. IEEE, 2008.

[88] Ravi, Srivaths, Raghunathan, Anand, and Chakradhar, Srimat. *Tamper resistance mechanisms for secure embedded systems.* In 17th International Conference on VLSI Design. Proceedings., pp. 605–611. IEEE, 2004.

[89] Rivest, Ronald L, Shamir, Adi, and Adleman, Leonard. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, 21(2):120–126, 1978.

[90] Roman, Rodrigo, Zhou, Jianying, and Lopez, Javier. *On the features and challenges of security and privacy in distributed internet of things.* Computer Networks, 57(10):2266–2279, 2013.

[91] Rose, Garrett S, McDonald, Nathan, Yan, Lok-Kwong, Wysocki, Bryant, and Xu, Karen. *Foundations of memristor based PUF architectures.* In Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures, pp. 52–57. IEEE Press, 2013.

[92] Rožić, Vladimir and Verbauwhede, Ingrid. *Hardware-Efficient Post-processing Architectures for True Random Number Generators.* IEEE Transactions on Circuits and Systems II: Express Briefs, 2018.

[93] Rukhin, Andrew, Soto, Juan, Nechvatal, James, Smid, Miles, and Barker, Elaine. *A statistical test suite for random and pseudorandom number generators for cryptographic applications.* Tech. rep., Booz-Allen and Hamilton Inc Mclean Va, 2001.

[94] Sanchez-Reillo, R, Sanchez-Avila, C, Lopez-Ongil, C, and Entrena-Arrontes, L. *Improving security in information technology using cryptographic hardware modules.* In Proceedings. 36th Annual 2002 International Carnahan Conference on Security Technology, pp. 120–123. IEEE, 2002.

[95] Santoro, Renaud, Sentieys, Olivier, and Roy, Sébastien. *On-line monitoring of random number generators for embedded security.* In 2009 IEEE International Symposium on Circuits and Systems, pp. 3050–3053. IEEE, 2009.

[96] Santoro, Renaud, Sentieys, Olivier, and Roy, Sébastien. *On-the-fly evaluation of FPGA-based true random number generator.* In 2009 IEEE Computer Society Annual Symposium on VLSI, pp. 55–60. IEEE, 2009.

[97] Schellekens, Dries, Preneel, Bart, and Verbauwhede, Ingrid. *FPGA vendor agnostic true random number generator.* In 2006 International Conference on Field Programmable Logic and Applications, pp. 1–6. IEEE, 2006.

[98] Schindler, Werner. *Evaluation criteria for physical random number generators.* In Cryptographic Engineering, pp. 25–54. Springer, 2009.

[99] Schindler, Werner. *Random number generators for cryptographic applications.* In Cryptographic Engineering, pp. 5–23. Springer, 2009.

[100] Schleiffer, Christian, Wolf, Marko, Weimerskirch, André, and Wolleschensky, Lars. *Secure key management-a key feature for modern vehicle electronics.* Tech. rep., SAE Technical Paper, 2013.

[101] Sehwag, Vikash and Saha, Tanujay. *TV-PUF: a fast lightweight analog physical unclonable function.* In 2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), pp. 182–186. IEEE, 2016.

[102] Shannon, Claude Elwood. *A mathematical theory of communication.* Bell system technical journal, 27(3):379–423, 1948.

[103] Sicari, Sabrina, Rizzardi, Alessandra, Grieco, Luigi Alfredo, and Coen-Porisini, Alberto. *Security, privacy and trust in Internet of Things: The road ahead.* Computer Networks, 76:146–164, 2015.

[104] Simons, P. *V. vanderLeest and E. vanderSluis,"Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs"*, 2012.

[105] Su, Ying, Holleman, Jeremy, and Otis, Brian. *A 1.6 pJ/bit 96% stable chip-ID generating circuit using process variations.* In 2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, pp. 406–611. IEEE, 2007.

[106] Suh, G Edward and Devadas, Srinivas. *Physical unclonable functions for device authentication and secret key generation.* In 2007 44th ACM/IEEE Design Automation Conference, pp. 9–14. IEEE, 2007.

[107] Sunar, Berk. *True random number generators for cryptography.* In Cryptographic Engineering, pp. 55–73. Springer, 2009.

[108] Sunar, Berk, Martin, William J, and Stinson, Douglas R. *A provably secure true random number generator with built-in tolerance to active attacks.* IEEE Transactions on computers, 56(1):109–119, 2006.

[109] Sustek, Laurent. *Hardware security module.* Encyclopedia of Cryptography and Security, pp. 535–538, 2011.

[110] Tao, Sha, Yu, Yang, and Dubrova, Elena. *FPGA Based True Random Number Generators Using Non-Linear Feedback Ring Oscillators.* In 2018 16th IEEE International New Circuits and Systems Conference (NEWCAS), pp. 213–216. IEEE, 2018.

[111] Tuyls, Pim and Škorić, Boris. *Physical unclonable functions for enhanced security of tokens and tags.* In ISSE 2006—Securing Electronic Busines Processes, pp. 30–37. Springer, 2006.

[112] Valtchanov, Boyan, Aubert, Alain, Bernard, Florent, and Fischer, Viktor. *Modeling and observing the jitter in ring oscillators implemented in FPGAs.* In 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, pp. 1–6. IEEE, 2008.

[113] Van Herrewege, Anthony, Katzenbeisser, Stefan, Maes, Roel, Peeters, Roel, Sadeghi, Ahmad-Reza, Verbauwhede, Ingrid, and Wachsmann, Christian. *Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs.* In International Conference on Financial Cryptography and Data Security, pp. 374–389. Springer, 2012.

[114] Varchola, Michal and Drutarovsk, Y. *New FPGA based TRNG principle using transition effect with built-in malfunction detection.* In International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices-CryptArchi. Prague (Czechia), pp. 150–155. Citeseer, 2009.

[115] Varchola, Michal and Drutarovsky, Milos. *New high entropy element for FPGA based true random number generators.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 351–365. Springer, 2010.

[116] Vasyltsov, Ihor, Hambardzumyan, Eduard, Kim, Young-Sik, and Karpinskyy, Bohdan. *Fast digital TRNG based on metastable ring oscillator.* In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 164–180. Springer, 2008.

[117] Veljković, Filip, Rožić, Vladimir, and Verbauwhede, Ingrid. *Low-cost implementations of on-the-fly tests for random number generators.* In 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 959–964. IEEE, 2012.

[118] Vernam, Gilbert S. *Cipher printing telegraph systems: For secret wire and radio telegraphic communications.* Journal of the AIEE, 45(2):109–115, 1926.

[119] Von Neumann, John. *13. various techniques used in connection with random digits.* Appl. Math Ser, 12(36-38):5, 1951.

[120] Walker, John. *Ent: A pseudorandom number sequence test program.* Software and documentation available at/www. fourmilab. ch/random/S, 2008.

[121] Wieczorek, Piotr Zbigniew. *Lightweight TRNG based on multiphase timing of bistables.* IEEE Transactions on Circuits and Systems I: Regular Papers, 63(7):1043–1054, 2016.

[122] Wold, Knut and Tan, Chik How. *Analysis and enhancement of random number generator in FPGA based on oscillator rings.* International Journal of Reconfigurable Computing, 2009:4, 2009.

[123] Yang, Bohan, Rožic, Vladimir, Grujic, Miloš, Mentens, Nele, and Verbauwhede, Ingrid. *ES-TRNG: A High-throughput, Low-area True Random Number Generator based on Edge Sampling.* IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 267–292, 2018.

[124] Yang, Yunfan, Jia, Song, Wang, Yuan, Zhang, Shaonan, and Liu, Chao. *A reliable true random number generator based on novel chaotic ring oscillator.* In 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4. IEEE, 2017.

[125] Yin, Chi-En Daniel and Qu, Gang. *LISA: Maximizing RO PUF's secret extraction.* In 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 100–105. IEEE, 2010.

[126] Zhun, Huang and Hongyi, Chen. *A truly random number generator based on thermal noise.* In ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549), pp. 862–864. IEEE, 2001.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] S. Buchovecká. *Testing a Random Number Generator.* POSTER 2011 - 15th International Student Conference on Electrical Engineering, Prague: CTU, Faculty of Electrical Engineering, 2011.

- ◦ Awarded by Program Committee of 15th International Student Conference POSTER 2011.

[A.2] S. Buchovecká, Josef Hlaváč. *Frequency injection attack on random number generator.* DDECS 2013 - IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, Karlovy Vary, Czech Republic, 2013.

The paper has been cited in:

- ◦ Toshifumi NISHINAGA and Masahiro MAMBO. *Implementation of uNaCl on 32-bit ARM Cortex-M0*, IEICE Transactions on Information and Systems, vol. E99.D, pp. 2056, 2016.
- ◦ Cicek, Ihsan, Mustafa Parlak, and Cetin Kaya Koc.*Light-weight FPGA Implementation of FIPS140-2 Online Statistical Tests*,14th International Workshop on Cryptographic Architectures Enbedded in Reconfigurable Devices (Crypt-Archi), Montpellier, France, 2016.
- ◦ Khas, A.A., Cicek, I.: *An Energy Efficient Run-Time Security Monitor for True Random Number Generators*, ELECO 2019 - 11th International Conference on Electrical and Electronics Engineering 8990560, pp. 488-491.
- ◦ Giechaskiel, Ilias and Rasmussen, Kasper Bonne. *Sok: Taxonomy and challenges of out-of-band signal injection attacks and defenses.* arXiv preprint arXiv:1901.06935 (2019).

- Chowdhury, S., Covic, A., Acharya, R. Y., Dupee, S., Ganji, F., Forte, D.: *Physical Security in the Post-quantum Era: A Survey on Side-channel Analysis, Random Number Generators, and Physically Unclonable Functions.* arXiv preprint arXiv:2005.04344 (2020).

[A.3] S. Buchovecká, R. Lórencz, F. Kodýtek, J. Buček. *True Random Number Generator based on ROPUF circuit.* DSD 2016 – Euromicro Conference on Digital System Design, Limassol, Cyprus, 2016.

The paper has been cited in:

- Latypov, R. Kh.; Stolov, E. L.: *Theory of ternary jitter-based true random number generators composed of identical gates*, Uchenye Zapiski Kazanskogo Universiteta, vol. 159, no. 2, 2017.
- Anahita, G., Krishnapriya, K.P.M., Shiva Prasad, R., Mohan Kumar, N.: *HD-Sign: Hardware based Digital Signature generation using true random number generator*, International Journal of Engineering and Technology(UAE) 7(3.8 Special Issue 8), pp. 147-150 (2018).
- Al-Haidary, Mohammed, and Qassim Nasir. *Physically Unclonable Functions (PUFs): A Systematic Literature Review*, 2019 Advances in Science and Engineering Technology International Conferences (ASET). IEEE, 2019.

[A.4] F. Kodýtek, R. Lórencz, J. Buček, S. Buchovecká. *Temperature dependence of ROPUF on FPGA.* DSD 2016 – Euromicro Conference on Digital System Design, Limassol, Cyprus, 2016.

The paper has been cited in:

- Kurra, A.K., Sadulla, S.: *Analysis of physical unclonable functions (PUFS) for secure key generation on smartcard chips*, Journal of Advanced Research in Dynamical and Control Systems 9(Special issue 14), pp. 1735-1745 (2019).
- Xu Jinfu, X.; Wu Jin, X.: *Frequency Sorting Algorithm Based on Dynamic Ring Oscillator Physical Unclonable Function Statistical Model*, Journal of Electronics & Information Technology, vol. 41, no. 3, 2019.
- Shaik, Sadulla, Anil Kumar Kurra, and A. Surendar. *High secure buffer based physical unclonable functions (PUF's) for device authentication*, Telkomnika 17.1 (2019).

[A.5] S. Buchovecká, R. Lórencz, F. Kodýtek, J. Buček. *True random number generator based on ring oscillator PUF circuit* . Microprocessors and Microsystems, 2017.

The paper has been cited in:

- Tuncer, S.; Kaya, T.: *True Random Number Generation from Bioelectrical and Physical Signals*, Computational and Mathematical Methods in Medicine, 2018.
- Aksshaya, B., G. Madhura, L.V., Nivethashri, S., Vishnuvarthini, T., Mohankumar, N.: *Design and analysis of analog TRNG using sample and hold circuit*, International Journal of Engineering and Technology(UAE) 7(3), pp. 69-73 (2018).
- Yakut, S.; Tuncer, T.; Ozer, A.: *Secure and Efficient Hybrid Random Number Generator Based on Sponge Constructions for Cryptographic Applications*, Elektronika ir Elektrotechnika, vol. 25, 2019.
- Hsiao, H., Anderson, J., Hara-Azumi, Y. *Generating stochastic bitstreams* Book Chapter), Stochastic Computing: Techniques and Applications pp. 137-152 (2019).
- Garipcan, A.; Erdem, E.: *Implementation of a Digital TRNG Using Jitter Based Multiple Entropy Source on FPGA*, Informacije MIDEM - Journal of Microelectronics, Electronic Components and Materials. , vol. 49, no. 2, 2019.
- Alibeigi, Iman, et al.: *A Low-cost Highly Reliable Spintronic True Random Number Generator Circuit for Secure Cryptography*, SPIN. World Scientific Publishing Company, 2019.
- Hsiao, Hsuan, Jason Anderson, and Yuko Hara-Azumi. *Generating Stochastic Bitstreams.* Stochastic Computing: Techniques and Applications, Springer, Cham, 2019.
- Etem, Taha, and Turgay Kaya. *A novel True Random Bit Generator design for image encryption*, Physica A: Statistical Mechanics and its Applications (2020)
- Boke, Amol K., Sangeeta Nakhate, and Arvind Rajawat. *Efficient Key Generation Techniques for Securing IoT Communication Protocols.* IETE Technical Review (2020): 1-12.
- Avaroglu, Erdinc *The implementation of ring oscillator based PUF designs in Field Programmable Gate Arrays using of different challenge.* Physica A: Statistical Mechanics and its Applications (2020): 124291.
- Garipcan, A.; Erdem, E.: *A TRNG using chaotic entropy pool as a post-processing technique: analysis, design and FPGA implementation*, Analog Integrated Circuits and Signal Processing 103(3), pp. 391-410 (2020).
- Etem, Taha, and Turgay Kaya. *A novel True Random Bit Generator design for image encryption.* Physica A: Statistical Mechanics and its Applications 540 (2020): 122750
- Boke, Amol K., Sangeeta Nakhate, and Arvind Rajawat. *Efficient Key Generation Techniques for Securing IoT Communication Protocols.* IETE Technical Review (2020): 1-12.
- Alibeigi, Iman, et al. *A Low-Cost Highly Reliable Spintronic True Random Number Generator Circuit for Secure Cryptography.* SPIN. Vol. 10. No. 01. World Scientific Publishing Company, 2020.

- Anandakumar, N. Nalla, Mohammad S. Hashmi, and Somitra Kumar Sanadhya. *Efficient and Lightweight FPGA-based Hybrid PUFs with Improved Performance.* Microprocessors and Microsystems (2020): 103180.
- Avaroğlu, Erdinç. *The implementation of ring oscillator based PUF designs in Field Programmable Gate Arrays using of different challenge.* Physica A: Statistical Mechanics and its Applications (2020): 124291.

[A.6] S. Buchovecká, R. Lórencz, J. Buček, F. Kodýtek. *Lightweight Authentication and Secure Communication Suitable for IoT Devices.* ICISSP 2020 - International Conference on Information Systems Security and Privacy, 2020.

- Paper shorlisted amongst selected papers from ICISSP 2020 Conference to be published in CCIS Series book published by Springer in extended and reviewed version.

Remaining Publications of the Author Relevant to the Thesis

[A.7] S. Buchovecká, *Hardware generated keys for cryptographic systems and protocols.* Postgradual Study Report, Faculty of Information Technology, Prague, Czech Republic, 2016.

[A.8] S. Buchovecká, J. Hlaváč, R. Lórencz. *New results in generating true random numbers on simple microcontrollers.* International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi), Ruhr Universität Bochum, 2011.

[A.9] F. Kodýtek, R. Lórencz, J. Buček, S. Buchovecká. *Multiple output bits ROPUF design for TRNG.* International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi), La Grande Motte, France, 2016.

The paper has been cited in:

- Habib, B.; Gaj, K.: *A comprehensive set of schemes for PUF response generation*, Microprocessors and Microsystems, vol. 51, 2017.

[A.10] S. Buchovecká, R. Lórencz, F. Kodýtek, J. Buček. *Secure authentication and communication for embedded systems using a PUF/TRNG combined circuit.* International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi), South Brittany, France, 2018.

[A.11] S. Buchovecká, R. Lórencz, F. Kodýtek, J. Buček. *Authentication and Secure Communication for Embedded Systems.* Work in Progress Session held in connection with the DSD 2019 – Euromicro Conference on Digital System Design, Kallithea, Chalkidiki, Greece, 2019.

Remaining Publications of the Author

[A.12] S. Buchovecká. *The future of financial services online access.* Information Security Summit (IS2), Prague, 2016.

[A.13] L. Kotlaba, S. Buchovecká, R. Lórencz, *Active Directory Kerberoasting Attack – Monitoring and Detection Techniques.* ICISSP 2020 - International Conference on Information Systems Security and Privacy, 2020.

- Paper received Best Poster Award Certificate at ICISSP 2020 Conference.