Chapter 3
Evaluation Criteria for Physical Random Number Generators

Werner Schindler

3.1 Introduction

In the previous chapter we first addressed general aspects of random number generators (RNGs) that are used for cryptographic applications. We divided the entirety of RNGs into two main classes, the deterministic RNGs (DRNGs) and the true RNGs (TRNGs), the latter falling into two subclasses, the physical RNGs (PTRNGs) and the non-physical true RNGs (NPTRNGs). Moreover, we distinguished between pure and hybrid RNGs, the latter deploying features from both, deterministic and true RNGs.

We formulated four general requirements ((R1) to (R4)). Which of these requirements are necessary depends on the concrete application. Inevitable for sensitive cryptographic applications are

- (R1) The random numbers should have good statistical properties.

  and

- (R2) The knowledge of subsequences 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 the subsequence.

For DRNGs, (R3) and (R4) (backward and forward secrecy) are essentially additional requirements. For TRNGs, (R3) and (R4) usually follow immediately from (R2) if we neglect very specific constructions.

In the previous chapter we treated DRNGs in detail, and we addresed basic facts of PTRNGs and NPTRNGs. The security of DRNGs essentially grounds on the complexity of the state transition function and the output function. In contrast, the security of TRNGs is based on 'true' randomness as their name indicates. DRNGs usually apply cryptographic primitives. Consequently, DRNGs can at the most be

Bundesamt für Sicherheit in der Informationstechnik
e-mail: Werner.Schindler@bsi.bund.de

Ç.K. Koç (ed.) *Cryptographic Engineering*, DOI 10.1007/978-0-387-71817-0_3,

computationally secure, and this assessment may change in course of time if weaknesses of the used primitives are detected and/or the computational power of a potential adversary increases. Unless the evaluator has made a mistake in the evaluation process, the determined workload to guess 'true' random numbers remains invariant in the course of the years. Consequently, it is reasonable to use TRNGs at least for the generation of random numbers that shall protect secrets in the long term. The connection between entropy and guessing workload was discussed in Section 5.2 of the previous chapter. In the last few years physical RNGs have attracted enormous attention in the scientific community and in the semiconductor industry. A large number of PTRNG designs have been proposed and analyzed ([4, 5, 7, 13, 25, 26, 42] etc.).

This chapter deals exclusively with the security evaluation of PTRNGs. The main task of the evaluator (but not his only one) is to determine the entropy per random bit, or at least a lower entropy bound. The evaluation of physical and non-physical TRNGs is very different. PTRNGs use dedicated hardware which is essentially identical for all devices (apart from tolerances of components or ageing effects). This allows a precise stochastic model of the noise source which is the basis of any sound entropy estimation. In contrast, NPTRNGs are usually implemented on PCs and exploit system data and/or the user's interaction. It is hardly possible to formulate a precise stochastic model that is valid for all implementations of the NPTRNG. Note that these implementations are not under the control of the designer of the NPTRNG (cf. Section 6 in Chapter 2).

At first we repeat central definitions and explain briefly the generic design of a physical RNG. The central goal of any PTRNG evaluation is the estimation of entropy per random bit. To reach this goal we develop general evaluation criteria and introduce the notion of a stochastic model. We investigate the algorithmic postprocessing, and online tests are also discussed in detail. Our expositions are illustrated by many examples and exercises. Further, we compare alternative security paradigms and have a brief look at the standards and evaluation guidances. We close the chapter with some thoughts on side-channel and fault attacks on PTRNGs.

3.2 Generic Design

Figure 3.1 illustrates the generic design of a physical RNG. The 'core' is the noise source, typically realized by electronic circuits (e.g., using noisy diodes or free-running oscillators) or by physical experiments (radioactive decay, quantum effects of photons, etc.) Usually, the noise source generates time-continuous analog signals which are (at least for electronic circuits usually) periodically digitized at some stage. We call the digitized values *digitized analog signals* or briefly *das random number*. Binary-valued das random numbers are also denoted as *das bits*. The das random numbers may be algorithmically postprocessed to internal random numbers in order to reduce potential weaknesses. The algorithmic postprocessing may be memoryless, i.e., it may only depend on the current das random numbers, or

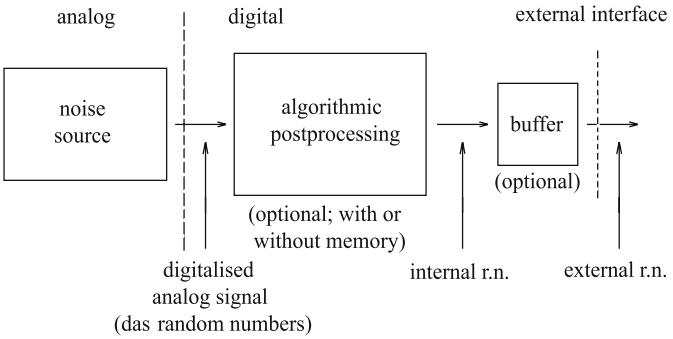

Fig. 3.1 Generic design of a physical RNG.

combine the current das random numbers with memory values that depend on the preceding das random numbers (and maybe some other, possibly secret parameters). Strong noise sources do not necessarily require algorithmic postprocessing. Upon external request internal random numbers are output (→ external random numbers).

3.3 Evaluation Criteria for the Principle Design

The primary goal of a PTRNG evaluation is the estimation of the entropy per random bit, or more precisely, the gain of entropy per random bit. A second task, which yet is not less important, is the evaluation of the online test (cf. Section 3.6). Depending on the intended conditions of use (and the concrete PTRNG design) it may be reasonable to implement explicit countermeasures against fault attacks. Such countermeasures (e.g., shielding) are clearly also part of an overall security evaluation (cf. Remark 3.10).

In Figure 3.1 three types of random numbers occur: das random numbers, internal random numbers and external random numbers. The first question is which of these random numbers should be considered. Of course, finally the quality of external random numbers is relevant since they are used in applications. However, the external random numbers are neither under the control of the RNG designer nor of the evaluator. Fortunately, the external random numbers are usually obtained by concatenating internal random numbers and hence share their statistical properties.

Unfortunately, entropy cannot be measured as voltage or temperature. Entropy is a property of random variables and not of sequences of observed values which are assumed by these random variables. As a first consequence, entropy cannot simply be guaranteed by applying any collection of statistical blackbox tests. Note that usually even pseudorandom sequences pass those blackbox test suites (cf. e.g., [27, 31, 32, 38] developed for a different purpose, namely for testing pseudorandom numbers for stochastic simulations), even if they have been generated by (in our sense) weak DRNGs.

Remark 3.1. The adjective 'universal' in the title of [28] caused a lot of misunderstanding and confusion in the past. Maurer's test segments large-bit streams into non-overlapping blocks of equal length. If the block size tends to infinity, Maurer's test value yields an estimator for the increase of entropy per random bit *provided that the random bits were generated by a stationary binary-valued ergodic random source with finite memory* (cf. also [11]).

If this assumption is not fulfilled Maurer's test value need not have any relation to the entropy per block. If the random numbers were generated by an LFSR, for instance, the increase of entropy per pseudorandom bit is obviously zero while Maurer's test value will be close to the maximum value. The same is true for Coron's test [10].

Instead, to quantify the entropy per random bit we first have to study the distribution of the random numbers, or more precisely, the distribution of the underlying random variables.

Definition 3.1. Random variables are denoted with capital letters. *Realizations* of these random variables, i.e., values assumed by these random variables, are denoted by the respective small letters. A binary-valued random variable is said to be $B(1,p)$-distributed if the values 1 and 0 are assumed with probability p and $1-p$, respectively. As usual, 'iid' abbreviates 'independent and identically distributed'.

The term $H(X)$ denotes the Shannon entropy of the random variable X while $H_\alpha(X)$ denotes its Rényi entropy to parameter α.

We interpret the das random numbers $r_1, r_2, \ldots$ as realizations of random variables $R_1, R_2, \ldots$ and the internal random numbers $y_1, y_2, \ldots$ as realizations of (suitably defined) random variables $Y_1, Y_2, \ldots$.

Remark 3.2. (i) For simplicity, we speak loosely of the entropy per random number or per random bit. What we really mean is the entropy of the underlying random variables.
(ii) In the following we use the term 'adversary'. This includes potential attackers but also the evaluator and the designer of an RNG.

In Section 5.2 of Chapter 2 we explained the relation between entropy and guesswork. We mentioned the min entropy is the most conservative entropy measure, which can be used to obtain universal lower guesswork bounds. However, if the random variables are iid or at least stationary with finite memory, the Shannon entropy quantifies the average guesswork at least asymptotically, i.e., for 'long' sequences of random numbers (e.g., session keys; cf. Chapter 2, Exercise 7). At least the internal random numbers are usually stationary with entropy per bit close to 1. Since the Shannon entropy is easier to handle than the min-entropy ($\rightarrow$ conditional entropy) we will focus on the Shannon entropy in the following. If it is unambiguous we follow the usual convention and briefly speak of 'entropy'.

The term

$$H(Y_{n+1} \mid Y_1 = y_1, \ldots, Y_n = y_n) \tag{3.1}$$

quantifies the entropy of the random variable Y_{n+1} if the adversary knows the internal random numbers $y_1,\dots,y_n$. This corresponds to the real-life situation where an adversary knows a subsequence $y_1,y_2,\dots,y_n$ of internal random numbers, e.g., from openly transmitted challenges or from the session keys of messages which he has received legitimately. The conditional entropy of consecutive internal random numbers (e.g., used to generate random session keys) equals

$$H(Y_{n+1},\dots,Y_{n+t} \mid Y_1 = y_1,\dots,Y_n = y_n) = \sum_{i=1}^{t} H(Y_{n+i} \mid Y_{n+1},\dots,Y_{n+i-1},Y_1 = y_1,\dots,Y_n = y_n). \tag{3.2}$$

Remark 3.3. (i) The conditional entropy (3.1) depends on the knowledge of the adversary (expressed by its conditional part) and quantifies his/her uncertainty on the next random variable.
(ii) For sensitive cryptographic applications (as for the generation of session keys, signature parameters or ephemeral keys) the conditional entropy per bit should be close to 1.
(iii) Note that entropy and conditional entropy may differ considerably. Assume, for instance, that X_1 is $B(1,0.5)$-distributed and $X_1 = X_2 = \cdots$ (total dependency). Then $H(X_n) = 1$ but $H(X_{n+1} \mid X_n) = 0$ for all $n \in \mathbb{N}$.

3.4 The Stochastic Model

The random variables $R_1,R_2,\dots$ and $Y_1,Y_2,\dots$ quantify the stochastic behavior of the das random numbers and the internal random numbers, respectively. These distributions clearly depend on the noise source and the digitization mechanism for the Y_j also on the algorithmic postprocessing algorithm. Usually, the evaluator is not able to determine these distributions exactly. Moreover, in a strict sense, the exact distribution depends on the components of the particular noise source which may differ to some extent even for PTRNGs from the same production series. Instead, as explained below, the evaluator shall specify a *family of distributions* and give evidence that the true distribution of the random numbers (as well as of specific auxiliary random variables) is always (i.e., for all copies of this PTRNG and under all conditions of use) contained in this family.

Example 3.1. (Repeated tossing with the same coin)
We interpret 'head' as 1 and 'tail' as 0, and we denote the generated (tossed) random numbers by $r_1,r_2,\dots$. For the moment we assume that no algorithmic postprocessing is applied, i.e., that $y_i = r_i$ for all i.

Since coins have no memory it is reasonable to assume that the random variables $R_1,R_2,\dots$ are iid binomially $B(1,p)$-distributed with unknown parameter $p \in [0,1]$. This means that the true distribution is contained in a one-parameter family of probability distributions (products of $B(1,p)$-distributions).

For a fixed coin an estimate $\widetilde{p}$ of the true parameter p can be achieved by tossing this coin N times and setting $\widetilde{p} := \#heads/N$.

For 'real-life' PTRNGs the situation is usually more complicated, and the specified family of distributions may depend on several parameters. It is usually reasonable to confirm the specified family of distributions by experiments. In Example 3.1 this may include tests for independency (although in this elementary example additional experiments seem to be superfluous). Generally speaking, the better the analog part of the PTRNG is understood the less experimental verification is necessary.

Our final goal is at least to determine a lower bound for the average entropy per internal random number. Ideally, the *stochastic model* of the noise source comprises the distribution of the internal random numbers, the das random numbers or at least the distribution of auxiliary random variables which allow one to calculate the entropy of the das random numbers or the internal random numbers, or at least to verify lower entropy bounds. We point out that a stochastic model is *not* a physical model of the noise source and its digitization mechanism.

In Example 3.1 a physical model would comprise the mass distribution within the coin and maybe complicated formulae that describe the trajectories of tossed coins (which would be a difficult task). Of course, the stochastic model also depends on the concrete noise source but considers only the impact on the distribution of the random numbers. In particular, we cannot expect that the stochastic model provides an explicit formula for the distribution of the das random numbers (and finally of the internal random numbers) in dependency of the characteristics of the components of the analog part of the noise source, which a physical model might achieve. Instead, we only get a *family of distributions* that depends on one or several parameters.

To reach our formulated goal, the estimation of entropy, we proceed as follows:

1. Formulate a stochastic model for the concrete PTRNG. Justify this model. Try to confirm your assumptions experimentally if there is no clear theoretical proof.
2. Use this PTRNG to generate random numbers. Estimate the unknown parameters that belong to this PTRNG on the basis of these random numbers.
3. Use these parameter estimates to estimate the increase of entropy per random bit.

Example 3.2. (Continuation of Example 3.1)
Recall that for repeated coin tossing the random variables $R_1, R_2, \ldots$ were assumed to be independent. If the sample size N in Example 3.1 was sufficiently large we conclude for any history $r_1, \ldots, r_n$

$$H(R_{n+1} \mid R_1 = r_1, \ldots, R_n = r_n) = H(R_{n+1}) \tag{3.3}$$
$$\approx -(\widetilde{p}\log_2 \widetilde{p} + (1-\widetilde{p})\log_2(1-\widetilde{p})) := \bar{H}(\widetilde{p}). \tag{3.4}$$

For small N (only a few coin tosses) it may be reasonable to determine a confidence interval $I(p)$ for the true parameter p (i.e., an interval that contains p with probability $\geq 1-\alpha$) and estimate the entropy for the least favorable parameter $p' \in I(p)$, providing a lower entropy bound for R_{n+1} (with probability $\geq 1-\alpha$).

Example 3.3. Assume that the random variables $R_1, R_2, \ldots$ form an ergodic homogeneous Markov chain on a finite state space $\Omega = \{\omega_1, \ldots, \omega_m\}$ with transition matrix $P = (p_{ij})_{1 \le i,j \le m}$ ([15], Section XV). As usual, $p_{ij} := \text{Prob}(R_{n+1} = \omega_j \mid R_n = \omega_i)$.

As $R_1, R_2, \ldots$ is an ergodic Markov chain, regardless of the distribution of R_1 the distributions of the random variables $R_1, R_2, \ldots$ converge exponentially fast to a unique limit distribution ν (the unique left eigenvector of P to the eigenvalue 1). If $p_{i.}$ denotes the ith row of P

$$H(R_{n+1} \mid R_1, \ldots, R_n) = H(R_{n+1} \mid R_n) = \sum_{i=1}^{m} \nu(i) H(p_{i.}), \tag{3.5}$$

at least in the limit $n \to \infty$. If the PTRNG is in equilibrium state when r_1 is generated, (which should be the case shortly after the start of the PTRNG), the random variables $R_1, R_2, \ldots$ may be assumed to be ν-distributed, and hence (3.5) is valid for all $n \in \mathbb{N}$ (see also Exercise 1).

Definition 3.2. A sequence of random variables $X_1, X_2, \ldots$ is called *stationary* if for any integer r the distribution of the random vector $(X_{t+1}, \ldots, X_{t+r})$ does not depend on the shift parameter t (cf. Remark 3.4). A sequence $X_1', X_2', \ldots$ is said to be q-dependent if the random vectors $(X_a', \ldots, X_b')$ and $(X_c', \ldots, X_d')$ are independent whenever $c - b > q$. The term $N(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2. The cumulative distribution function of the standard normal distribution $N(0,1)$ is denoted with $\Phi(\cdot)$, i.e., $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-0.5t^2}\, dt$

Remark 3.4. Some authors denote stochastic processes that meet the stationarity conditions from Definition 3.2 as *strictly stationary* to distinguish them from weakly stationary (aka weak-sense stationary or wide-sense stationary) stochastic processes. A stochastic process $X_1', X_2', \ldots$ is called weakly stationary if the expectations $E(X_j)$ and $E(X_j X_{j+\tau})$ do not depend on the index j.

For most PTRNG designs it is reasonable to assume that the sequence $R_1, R_2, \ldots$ is stationary, at least within time intervals that are large relative to the output rate of the PTRNG. (For example, if the PTRNG generates 1 million random bits per second, a minute or even an hour can doubtlessly be viewed as a 'long' time interval.) Long-term shifts of parameters in the life cycle of a PTRNG (e.g., due to ageing effects) are tolerable *if the distribution remains in the acceptable part of the specified family of distributions*. Note that it is usually very difficult to analyze non-stationary stochastic processes and, in particular, to get numerical results.

The distribution of $R_1, R_2, \ldots$ clearly determines the distribution of $Y_1, Y_2, \ldots$. However, complex algorithmic postprocessing may prevent concrete formulae for the Y_j. If the entropy per das random number is already sufficiently large, it may be sufficient to verify that the algorithmic postprocessing does not reduce the entropy per bit. Recall that in the end we are essentially interested in the entropy and not necessarily in the exact distribution of the Y_j (although the latter is cleary favorable).

Example 3.4. Assume that the random variables $R_1, R_2, \ldots$ are iid $B(1,p)$-distributed and that the algorithmic postprocessing algorithm encrypts consecutive, non-overlapping blocks of 128 das bits with a secret AES key k.

Since k is unknown, it is infeasible to specify the distribution of the random variables $Y_1, Y_2, \ldots$. However, the Y_j are iid, and since the postprocessing is injective,

$$H(Y_{n+1} \mid Y_n) = H(Y_{n+1}) = 128H(R_1). \tag{3.6}$$

(Of course, unless $p = 0.5$ the components of the Y_n are not independent.)

Although we are finally interested in the distribution (or at least in the entropy) of the random variables $Y_1, Y_2, \ldots$, we recommend generally to consider the random variables $R_1, R_2, \ldots$ first. The impact of the algorithmic postprocessing should be analyzed in a second step (see also Section 3.5). That is, in a first step we consider the conditional entropies

$$H(R_{n+1} \mid R_1 = r_1, \ldots, R_n = r_n) \text{ for any } r_1, \ldots, r_n, \text{ or at least} \tag{3.7}$$
$$H(R_{n+1} \mid R_1, \ldots, R_n). \tag{3.8}$$

The term (3.8) is the weighted average of (3.7) over all histories $r_1, \ldots, r_n$. Note that (3.8) is often easier to compute than (3.7). However, if (3.8) is close to the maximum value $\log_2(\text{range}(R_j))$, (3.7) may differ significantly from the average value of (3.8) only for a small fraction of histories. If non-negligible differences occur for different histories (or at least, if one expects such differences) it is reasonable to apply algorithmic postprocessing (see Section 3.5). This algorithmic postprocessing should compress the input data or at least 'mix' them.

Example 3.5. Figure 3.2 shows an RNG design that was proposed at CHES 2002 ([44]). In [42] a stochastic model was developed and analyzed, and a lower entropy bound for the internal random numbers was derived. Numerical examples were given. In the following section we address the central aspects.

The noise source consists of two independent ring oscillators that clock a 43-bit LFSR with a primitive feedback polynomial and a 37-bit linear cellular automaton shift register (CASR), respectively. The frequencies of the ring oscillators are not controlled and drift with variations in temperatures and voltage. The states of the

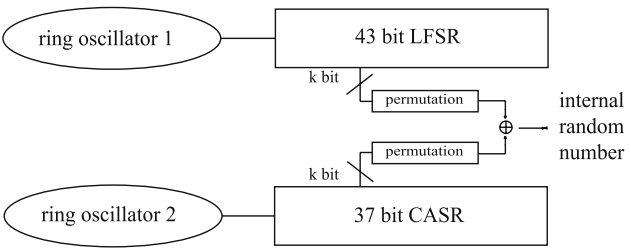

Fig. 3.2 Physical RNG presented at CHES 2002 ([44]).

LFSR and CASR are not reset after power-up. Upon external request $k = 32$ particular cells of both the LFSR and CASR are masked out, permutated and XOR-ed, and this 32-bit XOR-sum is output (internal random number). Even when no random numbers are requested the PTRNG is active. In [44] the minimum time between two consecutive outputs of internal random numbers is specified (LFSR and CASR should at least be clocked 86, and 74 times respectively).

Internal random numbers are output at times $s_0 < s_1 < \ldots$ where r_n is output at time s_n. The das random numbers are given by vectors $(r_{1(1)}, r_{1(2)}), (r_{2(1)}, r_{2(2)}), \ldots$ where $r_{j(1)}$ and $r_{j(2)}$ equals the number of cycles of ring oscillator 1 and 2, i.e., the number of clocks of the LFSR, as well as of the CASR, within the time interval $(s_{n-1}, s_n]$.

Since the oscillators are assumed to be independent we may treat them independently. (In a 'real' PTRNG evaluation the independence assumption requires a theoretical justification and eventually also experimental confimation.) Further, let $t_{i(1)}$ and $t_{i(2)}$ denote the lengths of the ith cycle of ring oscillator 1 and 2, respectively. In [42] we assumed that the corresponding random variables

$$T_{1(j)}, T_{2(j)}, \ldots \quad \text{are stationary} \tag{3.9}$$

(but not necessarily independent) for $j = 1, 2$. Further, let $z_{n(i)}$ denote the smallest index m for which $T_{1(i)} + T_{2(i)} + \ldots + T_{m(i)} > s_n$. With this notation

$$R_{n(i)} = Z_{n(i)} - Z_{n-1(i)} \qquad \text{for } i = 1, 2. \tag{3.10}$$

Example 3.6. Assume that the output voltage of a Zener diode is filtered, amplified and then input to a comparator. The comparator switches whenever the input voltage crosses a specified threshold from below (0-1-crossing). Each switching inverts the D input of a flip-flop. The flip-flop is latched by an external clock with constant period length $s > 0$.

We interpret the number of switchings in time interval $((n-1)s, ns]$ as das random number r_n. In this notation $y_n = y_0 + r_1 + \cdots + r_n \pmod 2$ where $y_0 \in \{0, 1\}$ is the internal random number at time $s = 0$. Let $t_1, t_2, \ldots$ denote the times between consecutive 0-1-switchings. Using the notation from Example 3.5 we conclude

$$R_n = Z_n - Z_{n-1}. \tag{3.11}$$

Also, for this example it is reasonable to assume that the random variables $T_1, T_2, \ldots$ are stationary.

Interestingly, although very different in their technical realization, Examples 3.5 and 3.6 lead to the same type of stochastic model for the das random numbers, and also an RNG construction with two noisy diodes which is intensively studied in [24] meets the following equations

$$T_1, T_2, \ldots \quad \text{are stationary} \tag{3.12}$$

$$R_n := Z_n - Z_{n-1} \quad \text{for} \tag{3.13}$$

$$Z_n := min_{m \in N}\{T_0 + T_1 + T_2 + \ldots + T_m > s_n\} \tag{3.14}$$

where T_0 denotes some 'offset' (the first switching, end of the first ring oscillator cycle etc. after time s_0). (In [42] we assumed $T_0 \equiv 0$ for simplicity, which had little impact in that scenario.) Note, however, that even if the stochastic models of two PTRNGs meet (3.12) to (3.14) the distributions of the random variables $T_1, T_2, \ldots$ and thus of $R_1, R_2, \ldots$ and $Y_1, Y_2, \ldots$ may yet be very different in both cases. Anyway, it is worth analyzing (3.12) to (3.14), both for very mild assumptions on the T_j and for specific classes of distributions (e.g., for iid T_j). We mention some fundamental results. For proofs, details and further assertions the interested reader is referred to [42] and [24].

For $u \geq 0$ we define

$$V_{(u)} := \inf\left\{\tau \in \mathbb{N} \mid \sum_{j=1}^{\tau+1} T_j > u\right\} = \sup\left\{\tau \in \mathbb{N} \mid \sum_{j=1}^{\tau} T_j \leq u\right\}. \tag{3.15}$$

Straightforward considerations lead to

$$\begin{aligned} &\mathrm{Prob}(V_{(u)} = k) = \mathrm{Prob}\,(T_1 + \cdots + T_k \leq u) - \mathrm{Prob}\,(T_1 + \cdots + T_{k+1} \leq u) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } k \geq 1 \\ &\mathrm{Prob}(V_{(u)} = 0) = 1 - \mathrm{Prob}\,(T_1 \leq u), \qquad \text{and} \qquad \mathrm{Prob}(V_{(u)} = \infty) = 0. \end{aligned} \tag{3.16}$$

Let $\mu := E(T_1)$ and $\sigma_T^2 := \mathrm{Var}(T_1) < \infty$. The term

$$\sigma^2 = \sigma_T^2 + 2\sum_{i=2}^{\infty} E\left((T_1 - \mu)(T_i - \mu)\right) \tag{3.17}$$

denotes the *generalized variance* of the random variables $T_1, T_2, \ldots$.

Assume further that $E|T_1^3| < \infty$. If the absolute values of the summands in (3.17) decrease 'rapidly' to zero then a version of the Central Limit Theorem for dependent random variables holds, i.e.,

$$\mathrm{Prob}\left(\frac{\sum_{j=1}^{k} T_j - k\mu}{\sqrt{k}\sigma} \leq x\right) \to_{k\to\infty} \Phi\left(\frac{x - k\mu}{\sqrt{k}\sigma}\right) \tag{3.18}$$

(for details, see [17, 21], for instance). This is in particular the case if the T_j are q-dependent since then the summands in (3.17) are identical zero for $i > q$. For $u = v\mu$ with $v >> 1$ we get the approximation

$$\mathrm{Prob}\left(V_{(v\mu)} = k\right) \approx \Phi\left(\frac{v-k}{\sqrt{k}} \cdot \frac{\mu}{\sigma}\right) - \Phi\left(\frac{v-(k+1)}{\sqrt{k+1}} \cdot \frac{\mu}{\sigma}\right) \quad \text{for } k \geq 1 \tag{3.19}$$

$$\mathrm{Prob}\left(V_{(v\mu)} = 0\right) \approx 1 - \Phi\left((v-1)\frac{\mu}{\sigma}\right). \tag{3.20}$$

By (3.19) and (3.20) the distribution of the random variable $V_{(v\mu)}$ depends only on the ratios μ/σ and $u/\mu = v$ but not on the absolute values of the parameters μ, σ^2, u. Note that the mass of $V_{(v\mu)}$ is essentially concentrated on those k's with $k \approx v$. Since

range$(V_{(u)}) = \mathbb{N}_0$, we extend the definition of Shannon entropy to countable Ω. For an $\mathbb{N}_0$-valued random variable X we define

$$H(X) = -\sum_{i=0}^{\infty} \text{Prob}(X=i)\log_2\left(\text{Prob}(X=i)\right) \quad . \tag{3.21}$$

We point out that $H(X) = \infty$ is possible ([42], Remark 2). In our context, the entropy is always finite (cf. Lemma 2(ii) in [42] which covers the q-dependent case). In particular, (3.19) and (3.20) imply

$$H(V_{(u)}) = -\sum_{i=0}^{\infty} \text{Prob}\left(V_{(u)=i}\right)\log_2\left(\text{Prob}\left(V_{(u)=i}\right)\right) \quad . \tag{3.22}$$

Intuitively, one expects that $H(V_{(u)})$ increases as u increases, although not necessarily monotonously. Note that if $\sigma^2 \approx 0$ then $H(V_{(k\mu)}) \approx 1$ but $H(V_{((k+0.5)\mu)}) \approx 0$ for small integers k, which explains the second statement. We may yet expect $H(V_{(u+\mu)}) > H(V_{(u)})$ since the positions of u and $u+\mu$ relative to the lattice points $\mu, 2\mu, \ldots$ are identical but more 0-1-crossings imply a larger variance of $V_{(\cdot)}$. Of course, this heuristic argumentation is not a strict mathematical proof (we work on it) but explicit computations for a large number of values u support our conjecture.

Since

$$R_n := \sup\left\{\tau \in \mathbb{N} \mid \sum_{j=1}^{\tau} T_{z_{n-1}+j} \le s_n - \sum_{j=1}^{z_{n-1}} T_j\right\} + 1 \tag{3.23}$$

the distribution of R_n is closely related to the distribution of the random variables $V_{(u)}$ considered above. Our goal is to determine (at least a lower bound for) the conditional entropy $H(R_{n+1} \mid R_1, \ldots, R_n)$. With regard to the preceding $H(V_{(s_{n+1}-s_n)})$ might be used as an estimate for this conditional probability, at least if the term $(s_{n+1}-s_n)/\mu$ (= expected number of T_j's within the interval $(s_{n+1}-s_n]$) is large and if $u \mapsto H(V_{(u)})$ only has little variation within an environment of $s_{n+1}-s_n$ (cf. Example 3.5 and [42]). Under these conditions one may hope that an eventual influence of $R_1, \ldots, R_n$ on (the first elements of) $T_{z_n+1}, T_{z_n+2}, \ldots$ has no significant impact on the entropy. To be on the safe side, one may try to (over-)compensate these effects by applying the more conservative entropy estimate

$$\min\{H(V_{(u)}) \mid u \in [s_{n+1}-s_n-a\mu, s_{n+1}-s_n]\} \cdot \text{Prob}(W_n \le a) \tag{3.24}$$

(with moderate $a > 0$) instead of $H(V_{(s_{n+1}-s_n)})$ where

$$W_n := T_0 + T_1 + \cdots + T_{z(n)} - s_n > 0 \quad . \tag{3.25}$$

Ideally, the parameter a should be selected with regard to the distribution of the T_j ($\rightarrow$ dependencies!) and the aimed entropy bound. At least for small ratios μ/s more sophisticated methods are recommendable that take the concrete distribution of the random variables T_j into account, since even moderate parameters a waste information. This demands larger interval lengths $(s_{n+1}-s_n)$ which in turn decreases the output rate of the PTRNG.

For a more sophisticated analysis we concentrate on equidistant instants $s_0, s_1, s_2, \ldots$, i.e., $s_n = ns$ for all $n \in \mathbb{N}$. Recall that the random variables $T_1, T_2, \ldots$ are assumed to be stationary (3.12). (This corresponds to the real-world situation that the noise source is in equilibrium state, which should be the case shortly after starting the PTRNG.) Under mild assumptions (essentially, the partial sums $(T_j + \cdots T_{j+\tau}) (\mathrm{mod}\, s)$ shall be uniformly distributed on $[0,s)$ for 'large' τ) it can be shown that the stochastic processes $R_1, R_2, \ldots$, $Y_1, Y_2, \ldots$, and $W_1, W_2, \ldots$ are stationary, too [24]. In particular, if G_W denotes the cumulative distribution function of W_n then

$$E((R_1 + \cdots + R_j)^k) = \int_0^{js} E((V_{(js-u)} + 1)^k \mid W_0 = u)) \, G_W(du) \tag{3.26}$$

$$\approx \int_0^{js} E((V_{(js-u)} + 1)^k) \, G_W(du) \quad \text{for each } k \in N. \tag{3.27}$$

For iid random variables T_j the $\approx$ sign is in fact $=$ while for dependent T_j the condition '$W_0 = u$' may influence the distribution of the first elements of the $T_1, T_2, \ldots$ via the conditional random variables $(\ldots, T_{-1}, T_0 \mid W_0 = u)$. If the T_j are Markovian, $(T_0 \mid W_0 = u)$ determines the initial distribution of the Markov process $T_0, T_1, \ldots$. Anyway, for 'large' indices j the influence of the condition '$W_0 = u$' on the integral should be negligible. Since $E(R_1 + \cdots + R_j) = jE(R_1)$ applying (3.27) for the parameters $(k = 1, j = 1)$ and, let's say, $(k = 1, j = 10)$ may serve as an indicator for the impact of W_0.

The stationarity of $R_1, R_2, \ldots$ implies

$$E((R_1 + \ldots + R_j)^2) = jE(R_1^2) + 2 \sum_{i=2}^{j} (j + 1 - i) E(R_1 R_i). \tag{3.28}$$

Computing the left-hand side for $j = 1, 2, \ldots$ with (3.26) or (3.27) yields $E(R_1)$ and $E(R_1 R_i)$ for $i = 1, 2, \ldots$, and finally the autocovariance function and the autocorrelation function of the stationary process $R_1, R_2, \ldots$. (We point out that the autocovariance function and the autocorrelation function are important quantities in the analysis of stochastic processes; cf. [45], for instance.) If the random variables $T_1, T_2, \ldots$ are iid mathematical renewal theory ([16], Chapter XI) yields a concrete formula for G_W. In particular, if the T_j have a continuous cumulative distribution function $G(\cdot)$ then $G_W(\cdot) = (1 - G(\cdot))/\mu$ and

$$H(R_{n+1} \mid R_0, \ldots, R_n) \geq \int_0^s H(V_{(s-u)}) \, \frac{1}{\mu} (1 - G(u)) \, du. \tag{3.29}$$

Proofs of the formulae mentioned above and a generalized version of (3.29) for which $G(\cdot)$ need not be continuous are given in [24], which also contains practical experiments. We mention that unless the ratio s/μ (as well as the ratio js/μ) is very small, the approximations (3.19) and (3.20) may be used to evaluate the integrals (3.26), (3.27) and (3.29).

Remark 3.5. (i) In the most general case (with not necessarily equidistributed instants s_n) the stochastic model consists of a family of distributions that belong to the auxiliary variables $V_{(u)}$ for $u \geq 0$. For any fixed u the parameters μ and σ define a two-parameter family of distributions. In combination with (3.24) this allows the coarse estimation of the conditional entropy $H(R_{n+1} \mid R_1, \ldots, R_n)$. The conditional entropy $H(Y_{n+1} \mid Y_1, \ldots, Y_n)$ clearly depends on the algorithmic postprocessing algorithm (see also Examples 3.12 and 3.13).
(ii) If $s_n = ns$ for all $n \in N$, the integrals (3.26), (3.27), (3.28) (and also (3.29) if the T_j additionally are iid) make the stochastic model more precise and comprehensive. The stochastic model then also comprises the autocovariance function of $R_1, R_2, \ldots$ and a sharper lower bound for the conditional entropy $H(R_{n+1} \mid R_1, \ldots, R_n)$.
(iii) Usually it is easier to derive a stochastic model for PTRNGs that exploit physical experiments (e.g., radioactive decay or quantum effects) than for PTRNGs that exploit electronic switchings (as in Examples 3.5 and 3.6). At least for smart cards the first type of PTRNGs is not relevant.

3.5 Algorithmic Postprocessing

In the last section our focus lay on the das random numbers $r_1, r_2, \ldots$ (or more precisely, on the underlying random variables $R_1, R_2, \ldots$). In this section we study the impact of the algorithmic postprocessing. For strong noise sources algorithmic postprocessing is not mandatory (cf. Section 3.6). Depending on the postprocessing algorithm, it may provide an additional security anchor. If the entropy per das random number is yet too low, a data-compressing postprocessing algorithm, which increases the average entropy per random bit, is an absolute must.

Example 3.7. Figure 3.3 shows a typical PTRNG design. The noise source generates single das bits $r_1, r_2, \ldots$ per time unit. In Step n bit r_n is XOR-ed to the feedback value of the LFSR, which is clocked synchronously to the digitization of the das bits. The right-most bit of the LFSR is the current internal random bit y_n.

If t denotes the number of LFSR cells for any initial state s_0 of the LFSR the mapping $(s_0, r_1 \ldots, r_n) \mapsto (y_{t+1}, \ldots, y_{t+n})$ is injective, implying

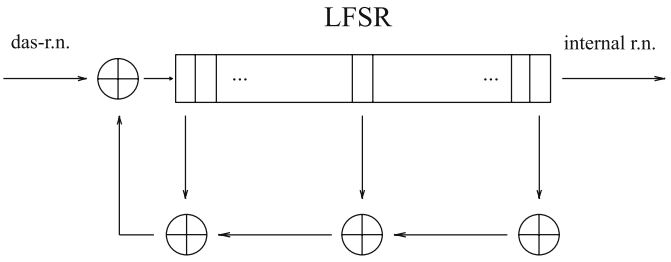

Fig. 3.3 An example of physical RNG with postprocessing.

$$H(R_1,\ldots,R_n) = H(Y_{t+1},\ldots,Y_{t+n} \mid s_0) \tag{3.30}$$

for any $n \in \mathbb{N}$. Asymptotically, this algorithmic postprocessing does not increase the average entropy per random bit, even if s_0 is unknown. It only transforms eventual weaknesses of the das random bits into others (e.g., bias into dependency). In the case of a total breakdown of the noise source the current state of the LFSR may serve as an 'entropy reserve'.

The analysis of the algorithmic postprocessing is more complicated if it shall increase the average entropy per random bit (data-compressing algorithms). In particular, its impact depends on the concrete distribution of $T_1, T_2, \ldots$.

Example 3.8. Assume that the random variables $R_1, R_2, \ldots$ are iid $B(1,p)$-distributed.
(a) Divide the sequence $R_1, R_2, \ldots$ into non-overlapping pairs $(R_1,R_2),(R_3,R_4),\ldots$ and define $Y_n := R_{2n-1} + R_{2n} \pmod 2$. If $p = 0.5 + 0.5\varepsilon$ with $\varepsilon \in [-1,1]$ elementary calculations verify that the internal random numbers are iid $B(1,p')$-distributed with $p' = 0.5 - 0.5\varepsilon^2$. For small ε we have $\log_2(0.5+0.5\varepsilon) \approx 1 - \varepsilon/\log(2)$ (linear Taylor expansion). Consequently, this algorithmic postprocessing increases the entropy per random bit from $1-\varepsilon^2/\log(2)$ to $1-\varepsilon^4/\log(2)$.
(b) (von Neumann transformation; cf. e.g., [30]) Divide the sequence $R_1, R_2, \ldots$ into non-overlapping pairs $(R_1,R_2),(R_3,R_4),\ldots$. Discard the pair (R_{2n-1},R_{2n}) if $R_{2n-1} = R_{2n}$, otherwise, the output is taken as R_{2n-1}. The internal random numbers are iid $B(1,0.5)$-distributed (ideal RNG!). In particular, $H(Y_{n+1} \mid Y_1,\ldots,Y_n) = H(Y_{n+1}) = 1$.

We point out that both postprocessings from Example 3.8 reduce the output rate of the RNG; Variant (a) by constant factor 2 while Variant (b) reduces the output by factor $p(1-p) \le 0.25$, depending on the parameter p. Both variants increase the quality of the random numbers, von Neumann's transformation even produces sequences of ideal random numbers *provided that the* $R_1, R_2, \ldots$ *are iid*. We refer the interested reader to Peres' unbiasing algorithm [34] which exploits the same idea as von Neumann's algorithm. At the cost of complexity, Peres' algorithm increases the output rate. The following example considers the impact of postprocessing Variant (a) on Markovian random variables $R_1, R_2, \ldots$ (cf. Exercises 3 and 4).

Example 3.9. We consider the algorithmic postprocessing from Example 3.8(a) but we assume that the binary-valued random variables $R_1, R_2, \ldots$ form an ergodic homogeneous Markov chain with transition matrix $P = (p_{ij})_{0\le i,j\le 1}$ (cf. Example 3.3). In particular, there exists a stationary limiting distribution ν.
We point out that the random variables $Y_1, Y_2, \ldots$ are in general no longer Markovian, which complicates the exact analysis of the entropy. If we assume that the random variables $R_1, R_2, \ldots$ are already in equilibrium state, straightforward considerations yield

$$\begin{aligned} &H(Y_{n+1} \mid Y_1,\ldots,Y_n) \ge H(Y_{n+1} \mid R_1,\ldots,R_{2n}) = H(Y_{n+1} \mid R_{2n}) = \\ &H(R_{2n+1} \oplus R_{2n+2} \mid R_{2n}) = \bar{H}(\nu(0)p_{01}(p_{10}+p_{00}) + \nu(1)p_{10}(p_{11}+p_{01})) = \\ &2p_{01}p_{10}/(p_{01}p_{10}) \ . \end{aligned} \tag{3.31}$$

On the other hand, $H(R_{2n+1} \mid R_{2n}) = \bar{H}(\nu(0)p_{01} + \nu(1)p_{11})$. We point out that the lower entropy bound from (3.31) can be further improved (cf. Exercise 3).

Example 3.10. (One-way postprocessing)
(a) A dedicated hash function $\mathrm{Ha}(\cdot)$ with output length m bits (e.g., Ha=SHA-1 with $m = 160$ or Ha=SHA-256 with $m = 256$) is applied to non-overlapping blocks $b_1, b_2, \ldots$ of consecutive $(m+k)$ das bits ($k \geq 0$). The hash values are used as internal random numbers.
(b): As in (a), but the previous internal random number (extended by k zeroes) and the current das bit block are first XOR-ed to an $(m+k)$-bit register, and then the hash function is applied to this register.

At first sight both variants may seem to be equivalent. As far as the PTRNG generates strong das random numbers this is indeed true. The second variant is yet preferable if the quality (entropy) of the das random numbers decreases (which yet should be detected by the online test!, see Section 3.6). Note that even if the PTRNG produces constant (known) sequences of das random numbers variant (b) still constitutes a strong DRNG which fulfils requirement (R3) (see Chapter 2) provided, of course, that the $(m+k)$ bit register may be viewed as uniformly distributed. This is the case if at least one 'strong' das bit block had been added to the register or several XOR-operations with low entropy blocks have been performed.

Example 3.11. (Continuation of Example 3.10)
It is practically infeasible to characterize the distribution of the internal random numbers or to calculate its entropy explicitly. On the other hand, hash functions are commonly assumed to behave in many regards as random mappings. We determine below the mean entropy for a related scenario where the algorithmic postprocessing algorithm is selected independently and uniformly from the set $\mathscr{F}_{(m+k,m)} := \{\phi\colon \{0,1\}^{m+k} \to \{0,1\}^m\}$ for each internal random number. The randomly selected algorithmic postprocessing algorithms are assumed to be publicly known. For simplicity, we assume that the das random bits are iid $B(1, 0.5)$ distributed. Consequently, the random input blocks $B_1, B_2, \ldots$ are iid uniformly distributed on $\{0,1\}^{m+k}$ for both variants. Symmetry implies

$$
\begin{aligned}
&E\left(H(\mathrm{Ha}(B_n))\right) = \qquad (3.32)\\
&-\frac{1}{|\mathscr{F}_{(m+k,m)}|} \sum_{\phi \in \mathscr{F}_{(m+k,m)}} \sum_{y \in \{0,1\}^m} \frac{|\phi^{-1}(y)|}{2^{(m+k)}} \log_2 \left(\frac{|\phi^{-1}(y)|}{2^{(m+k)}} \right) =\\
&-\frac{2^m}{|\mathscr{F}_{(m+k,m)}|} \sum_{\phi \in \mathscr{F}_{(m+k,m)}} \frac{|\phi^{-1}(y_0)|}{2^{(m+k)}} \log_2 \left(\frac{|\phi^{-1}(y_0)|}{2^{(m+k)}} \right)
\end{aligned}
$$

where $y_0 \in \{0,1\}^m$ is arbitrary but fixed. To simplify notation we temporarily use the abbreviations $v := 2^{m+k}$ and $w := 2^m$. For any fixed subset $J \subseteq \{0,1\}^{m+k}$ the probability of being the pre-image of a uniformly selected mapping $\phi \in \mathscr{F}_{(m+k,m)}$ clearly is $(w^{-1})^{|J|}(1-w)^{v-|J|}$. Hence the last term of (3.32) equals

$$-w\sum_{j=0}^{v}\binom{v}{j}\left(\frac{1}{w}\right)^{j}\left(\frac{w-1}{w}\right)^{v-j}\left(\frac{j}{v}\right)\log_2\left(\frac{j}{v}\right)= \tag{3.33}$$

$$\frac{-w}{v}\left[\sum_{j=0}^{v}\binom{v}{j}\left(\frac{1}{w}\right)^{j}\left(\frac{w-1}{w}\right)^{v-j}j\log_2(j)-\log_2(w)\sum_{j=0}^{v}\binom{v}{j}\left(\frac{1}{w}\right)^{j}\left(\frac{w-1}{w}\right)^{v-j}j\right]$$

$$=\log_2(w)-\frac{w}{v}\sum_{j=0}^{v}\binom{v}{j}\left(\frac{1}{w}\right)^{j}\left(\frac{w-1}{w}\right)^{v-j}j\log_2(j).$$

The second term quantifies the 'entropy defect' from the uniform distribution on $\{0,1\}^m$. The DeMoivre-Laplace approximation yields

$$\log_2(w)-\frac{w}{v}\sum_{j=0}^{v}\frac{\exp^{\frac{-(j-v/w)^2}{2v(1/w)(w-1)/w}}}{\sqrt{2\pi(v/w)(w-1)/w}}j\log_2(j)\approx \tag{3.34}$$

$$\log_2(w)-2^{-k}\sum_{j=0}^{v}\frac{exp^{\frac{-(j-2^k)^2}{2^{k+1}}}}{\sqrt{2\pi 2^k}}j\log_2(j)\approx$$

$$\log_2(w)-\frac{2^{-k}}{\log(2)\sqrt{2\pi 2^k}}\int_0^{\infty}\exp^{\frac{-(t-2^k)^2}{2^{k+1}}}t\log(t)\,dt.$$

(For small integer k the Poisson approximation may give better approximations.)

Note that the integrand is $\leq \exp^{\frac{-(j-2^k)^2}{2^{k+1}}}t^2$. Extending the domain of integration from $[0,\infty)$ to $(-\infty,\infty)$ provides the coarse lower entropy bound $\log_2(w)-1/((1+2^k)\log(2))$. Note that any computer algebra system enables the precise evaluation of (3.34) (Exercise 5).

Example 3.12. (Continuation of Example 3.5)
The stochastic model that was specified in (3.12) to (3.14) fits to the das random numbers from Example 3.5; see also [42], 'Justification of the stochastic model'.

The randomness of the internal random numbers follows from the uncertainty about how often the LFSR and CASR are clocked between successive outputs of internal random numbers. In contrast, if an adversary knows the number of clocks, the internal random numbers (and unpublished design parameters), he can form a system of equations to determine the internal state of LFSR and CASR, computing the next internal random numbers from the number of cycles between the next outputs.

The average increase of entropy per internal random number (k bits) is clearly limited from above by the average increase of entropy per das random number within this time period. It is yet not obvious how much entropy are really extracted by XORing k selected bits from the LFSR and CASR. In [42] an upper and a lower entropy bound for the internal random numbers is worked out. For details we refer the interested reader to [42], in particular to Theorems 1 and 2. Numerical results are collected in [42], Table 3.1. For $k=1$, $\mu/\sigma=0.01$ and time $s=60,000\cdot\mu$ between successive outputs of internal random numbers. For instance, we have $H(Y_{n+1}\mid Y_1,\ldots,Y_n)\geq 0.991$, while $H((R_{n+1(1)},R_{n+1(2)})\mid$

$(R_{1(1)},R_{1(2)}),\ldots,(R_{n(1)},R_{n(2)})) = 6.698$, providing an upper entropy bound for the internal random numbers. Note that this upper entropy bound does not provide a positive result on the question on how large k may be chosen. It yet says that in no case more than $k = 6$ bits should be extracted. For $s = 10,000 \cdot \mu$, we obtain $H(Y_{n+1} \mid Y_1,\ldots,Y_n) \geq 0.943$ for $k = 1$, (respectively, we have $H \geq 1.827$ and $H \geq 2.600$ for $k = 2$ and $k = 3$). The average entropy per das random number (two ring oscillators) then is 4.209 bit.

Example 3.13. As already pointed out, the stochastic properties of das random numbers in Example 3.6 can also be described. with the stochastic model that was specified in (3.12) to (3.14). The postprocessing is less complicated than in Example 3.12. Simplify speaking, we essentially have to replace $V_{(s-u)}$ by $V_{(s-u)} \pmod 2$ (see [24] for details).

Remark 3.6. As already pointed out in Chapter 2, Remark 2.1, for particular cryptographic applications non-complex relations exist that provide information on the unknown random numbers. For instance, N DSA- or ECDSA signatures yield a system of N linear equations in $N+1$ variables. Any disclosed ephemeral key immediately compromises the signature key x and, similarly, if an adversary knows a few bits from many ephemeral keys more sophisticated lattice-based methods also allow one to recover x (see, e.g., [18]). If the signature key x and all ephemeral keys yet were generated by an ideal RNG, these linear equations would not provide any additional information on x (neglecting other information as the public key, for instance (→ discrete log problem)); finding e.g., a DSA-signature key still would remain a 160 problem. Principally, even a small entropy defect per internal random bit may violate this 'theoretical' security property of the linear equations if N is sufficiently large (neglecting any other information on x), although it should be noted that no practial attack is known that exploits (in particular, small) entropy defects that are 'smeared' over the randomly selected ephemeral keys. If the das random numbers, as well as the internal random numbers, are iid, the von Neumann transformation (Example 3.8b) may be applied to eliminate eventual entropy defects. In the general case, it might be an option to apply an additional strong cryptographic postprocessing algorithm on the internal random numbers for applications of this type, providing a second security anchor (cf. Section 3.7).

3.6 Online Test, Tot Test, and Self Test

In Sections 3.3, 3.4, and 3.5 we discussed intensively how to estimate the entropy per random number. We introduced the concept of a stochastic model and illustrated the evaluation process by several examples. We specified and analyzed several parameter-dependent families of distributions (→ stochastic model), and we considered the impact of the algorithmic postprocessing. If a single PTRNG is concerned, the parameter(s) are estimated from random number sequences that were generated with this device. When evaluating a PTRNG design that is implemented on millions of smart cards the parameter(s) are estimated only for a few PTRNGs.

However, a concrete PTRNG in operation may generate random numbers that are much weaker than those of the carefully investigated prototypes in the laboratory. This may have several reasons, e.g., tolerances of components, ageing effects, external influences (fault attacks) or even a total failure of the noise source.

Requirement (O1) characterizes the task of a *tot test* ('tot' stands for 'total failure', not for the German adjective tot (=dead)).

- (O1) The tot test should detect a total breakdown of the noise source almost immediately.

More precisely, the tot test shall detect a total breakdown before any internal random number can be output that was influenced by das random numbers which were generated after a total failure of the noise source. (A total failure reduces the entropy of the following das random numbers to 0.) This requirement may be relaxed for memory-dependent algorithmic postprocessing algorithms since the history variables constitute an 'entropy reserve' (cf. Example 3.7) In the best case a total breakdown causes a constant sequence of das random numbers, which can easily be detected. Depending on the noise source more complicated pseudorandom patterns may occur; consider for instance the stochastic model specified in (3.12) to (3.14). The reduction of the generalized variance σ^2 to zero implies the generation of non-constant pseudorandom sequences. The tot test may be realized by a statistical test or by measurements of technical parameters such as electrical current, capacity, etc.

A *self test* should be applied after each start of the PTRNG, just to check the functionality in a qualitative sense. Self tests need not meet specific requirements.

The task of the *online test* is most critical. It shall detect any *non-tolerable* weaknesses of the random numbers while the RNG is in operation. The next section is exclusively devoted to online tests.

Besides the entropy analysis of the concrete design (inclusive the algorithmic postprocessing), the evaluator also has to verify the effectiveness of the online, tot and self tests. This also comprises the specified consequences of noise alarms, i.e., when a statistical test fails.

Remark 3.7. The literature does not consistently distinguish between online test, tot test and self test. Some authors speak of online tests or health tests, which include the tasks of the tot test and the self test. In our understanding, these tests should be distinguished at least from a logical point of view since their tasks are different. However, in concrete implementations, a single test may cover all aspects simultaneously.

3.6.1 Online Tests

The first question clearly is which random numbers should be tested. Since the external random numbers are not under the designer's control we only have the choice between the das random numbers and the internal random numbers.

Let us reconsider Example 3.7. In the worst case the noise source totally breaks down, generating a constant sequence of das bits. For a constant sequence $\ldots,0,0,\ldots$ of das random numbers the PTRNG is equivalent to a free-running LFSR which is a weak DRNG; the internal random numbers are then deterministic, having entropy 0. However, if the LFSR is not too short, the internal random numbers $y_1, y_2, \ldots$ will pass almost any collection of statistical blackbox tests unless specific characteristics of LFSR output sequences (as the linear complexity profile) are tested. Serious weaknesses of the noise source (though not a total breakdown) can hardly be detected by statistical tests. Testing the das bits would reveal at least a total breakdown immediately.

This elementary example points to a general rule, namely that usually the das bits should be tested if the RNG permits access. We point out that in very specific situations it may also be reasonable to test the internal random numbers instead. This may be the case if the das random numbers only possess low entropy and if their distribution may assume very different set parameters, e.g., if we consider a whole production series. Of course, effective tests for internal random numbers demand precise stochastic models of the internal random numbers, which seems feasible at the most for simple postprocessing algorithms (cf. Example 3.8). In contrast, Example 3.10 provides a typical counterexample. Even low entropy input causes statistically inconspicious output random numbers.

We point out that there do not exist statistical tests that are universally strong for any PTRNG design. For iid $B(1,p)$-distributed $R_1, R_2, \ldots$ (cf. Example 3.1), for instance, a monobit test which simply counts the number of zeroes and ones within a sample, is clearly appropriate. On the other hand, weaknesses of the type $\ldots,0,1,0,1,\ldots$ will not be detected by a monobit test.

Instead, statistical tests should be tailored to the stochastic model of the das random numbers or the internal random numbers (cf. Example 3.14). The distribution of $R_1, R_2, \ldots$, as well as of $Y_1, Y_2, \ldots$, and of auxiliary random variables shall remain in the class of distributions that was specified in the stochastic model under all circumstances, in particular, if non-tolerable weaknesses of the R_j occur.

Requirement (O2) to requirement (O4) formulate generic properties that online tests should fulfil.

- (O2) Non-tolerable statistical weaknesses of the das random numbers should be detected sufficiently fast.
- (O3) If the weaknesses of the random numbers are tolerable (i.e., if the 'distance' of the respective random variables from iid uniformly distributed random variables ($\rightarrow$ ideal RNG) is small) the probability for a noise alarm should be (almost) negligible.
- (O4) The online test should run fast and require only a few lines of code and little memory.

Remark 3.8. 'Real-world' PTRNGs are hardly optimal, at least not in a strict sense. However, certain deviations from an ideal RNG are tolerable. To ensure functionality, the online test should be passed with overwhelming probability if the weaknesses are tolerable ($\rightarrow$ (O3)).

Example 3.14. (i) In Example 3.1 we assumed that the das random numbers are iid $B(1,p)$-distributed. For this stochastic model, a monobit test would be appropriate.
(ii) Example 3.3 we considered a two-parameter family of distributions which may be parametrized by the transition probabilities p_{01} and p_{10}. Equality $p_{01} = p_{10} \in (0,1)$ implies $\nu = (0.5, 0.5)$. Although the random variables $R_1, R_2, \ldots$ may be far from being independent (if $p_{01} = p_{10} = 0.8$, for instance) it is very likely that the monobit test will be passed. Hence the monobit test is not effective for this PTRNG. Effective online tests should consider transition probabilities (see Exercise 6).
(iii) In Examples 3.5 and 3.12 (as well as in Examples 3.6 and 3.13) the situation is more complicated. An effective test of the internal random numbers seems to be hardly possible. Instead, the das random numbers should be tested. For time intervals with given length u, the increase of entropy essentially depends on μ and σ^2. Hence it is natural to check the arithmetic mean and the empirical variance of the number of clock cycles (as well as switchings) within time intervals of fixed length ($\rightarrow$ Exercise 7).

Statistical tests with extremely small rejection probabilities are widely spread in practice. The following example illustrates the disadvantages of those approaches. For details we refer the interested reader to [40], Section 4.

Example 3.15. The binary-valued das random numbers $r_1, \ldots, r_{320}$ are grouped into 80 non-overlapping 4-bit words which are identified with integers from 0 to 15. A χ^2-goodness-of-fit test on these 4-bit words (aka poker test) is applied

$$c := \sum_{i=0}^{15} \frac{(fr(i) - 5)^2}{5} \tag{3.35}$$

where $fr(i)$ denotes the number of i's within the eighty 4-bit words ([22], p. 69). The test is passed if the test value $c \leq 65$.

If the $R_1, R_2, \ldots$ are iid $B(1, 0.5)$-distributed the χ^2-test variable C is multinomially distributed. *If the sample size tends to infinity* the distribution of C tends to the χ^2-distribution with 15 degrees of freedom, χ^2_{15}. For moderate rejection probabilities this approximation is fully appropriate. However, $\text{Prob}(X > 65.0) = 3.4 \cdot 10^{-8}$ for a χ^2_{15} distributed random variable X, while exact computations yield $\text{Prob}(C > 65.0) = 3.8 \cdot 10^{-7}$. Although the absolute error is small the relative error is not, namely

$$\frac{|\text{Prob}(T > 65.0) - \text{Prob}(X > 65.0)|}{\text{Prob}(X > 65.0)} \approx 10.1 \ . \tag{3.36}$$

Note that for the rejection boundary 30.6, for instance, the relative error is only ≈ 0.03.

This example points to a general problem: Often, only the limit distribution of the test variable is known (i.e., when the sample size of the statistical test converges to infinity), usually only for one specific distribution (typically for iid uniformly distributed random variables). The relative approximation error at the tails of the distribution is usually considerably large. In Example 3.15 even for an ideal RNG,

10 times more online tests would fail than the designer expects if he relies upon the χ^2-approximation. This is not a security problem but may affect functionality. Other statistical tests may show the opposite behavior, i.e., less failures of the online test may occur than expected, which might induce security problems.

Another relevant question is the failure probability of the online test for random numbers that are not ideal. The limit distribution does not give an answer even for moderate rejection probabilities α (e.g., $\alpha \approx 10^{-3}$). However, moderate α allows approximations by stochastic simulations ([40], Section 7; see also Chapter 2, Section 4.3). According to a specified distribution (which is relevant for the PTRNG in evaluation), pseudorandom numbers $\widetilde{b}_1, \widetilde{b}_2, \ldots$ are generated. For example, iid $B(1, 0.485)$-distributed random variables or a Markov chain with a particular transition matrix P may be simulated. The statistical test is applied to the pseudorandom numbers in place of the das random numbers, as well as of the internal random numbers. Repeating this process many times gives an empirical cumulative distribution which provides estimates for particular rejection probabilities under the specified distribution (e.g., $\text{Prob}(C > 30.6)$ in Example 3.15 for iid $B(1, 0.485)$-distributed random variables). To obtain reliable estimates for the unknown rejection probabilities, as a rule of thumb we recommend to repeat the simulations at least $M \geq 100/\alpha$ times where α denotes the true but unknown rejection probability of interest. (Somewhat more than 100 failures of the basic test indicate that this number M should be reached.) Extremely small rejection probabilities as 10^{-9}, for instance, require gigantic workload which is a further argument to consider moderate rejection probabilities.

The key idea is to apply an elementary statistical test ($\rightarrow$ (O4)) to the random numbers but to exploit the test value by different rules which cover the requirements formulated in (O1) to (O3) above. We sketch the procedure that was introduced in Ref. [40]. We recommend the interested reader to study this reference.

At first the designer selects a statistical test, the so-called *basic test*, with regard to the specific PTRNG (as well as with regard to its stochastic model). This might be a monobit test ($\rightarrow$ Example 3.1), a test that considers one-step transition frequencies ($\rightarrow$ Example 3.3) or a χ^2 goodness of fit test, for example, provided that the required memory, the lines of code and the execution time are acceptable for the device used and the intended applications. Note, however, that the particular choice of the basic test does not affect the general principle of the online test procedure.

A *test suite* consists of at the most N basic tests. The basic test values are denoted with $c_1, c_2, \ldots$ while $hc_0, hc_1, hc_2, \ldots$ denote the *history variables*. We start with $hc_0 := \text{E}_{0,r}(C)$, the expectation of the basic test for ideal RNGs, rounded to a multiple of 2^{-c}. In Step $j \in \{1, \ldots, N\}$ a basic test is performed, and $hc_j := (1-\beta)hc_{j-1} + \beta c_j$ is computed (with $\beta = 2^{-b} \ll 1$) and rounded to a multiple of 2^{-c} where c denotes a fixed integer (typically $c \in \{5, 6\}$). Since β is a power of 2, updating the history variable only needs integer arithmetic. In Step j the following decision rules are applied:

(A): if $c_j \notin S_{(A)} \Rightarrow$ stop the test suite + noise alarm
(B): if $c_{j-k+1}, \ldots, c_j \notin S_{(B)} \Rightarrow$ stop the test suite + noise pre-alarm
(C): if $hc_j \notin S_{(C)} \Rightarrow$ stop the test suite + noise pre-alarm

If no noise alarm or noise pre-alarm occurs within the steps $1,\ldots,N$ a new test suite begins. After a noise pre-alarm a new test suite begins. Noise pre-alarms in $x \geq 1$ consecutive test suites induce a noise alarm. The designer of the PTRNG should specify the consequences of a noise alarm in the user's manual of the PTRNG (if there is any). The consequences should be adjusted to the concrete PTRNG design, the basic test, the decision rules (A) to (C), the type of application, etc. The most restrictive consequence is to stop the generation of the random numbers forever. Alternatively, the generation of further random numbers may be accepted after a specific test procedure has been passed, or a manual restart of the TRNG may be permitted. In these cases, noise alarms should be logged.

Criterion (A) covers the tot test functionality. The set $S_{(A)}$ is selected that a failure of criterion (A) is extremely unlikely for any acceptable distributions of the random numbers. Criterion (B) combines several consecutive failures of the basic test, each of them occurring with probability between 10^{-3} and 10^{-2} (to give a rule of thumb). The history variables $hc_1, hc_2, \ldots$ shall detect if the mean value of the test values $c_1, c_2, \ldots$ drifts too far from $E_{0,r}(C)$ without increasing the sample size of the basic test (for details see [40]). Note that single statistical tests may be viewed as a special case where the complements of $S_{(A)}$ and $S_{(C)}$ are empty (i.e., no condition) and $k = 1$.

In order to select appropriate sets and parameters the range of possible distributions ($\rightarrow$ stochastic model) of the das random numbers, as well as of the internal random numbers, and of auxiliary random variables, shall be divided into three subsets, possibly under consideration of the intended applications: the subset of distributions which are fully agreeable, the subset of non-tolerable distributions and the complement of these two subsets. If the distribution of the random numbers lies in the first subset a noise alarm should occur only with negligible probability, while a noise alarm should occur as soon as possible if the distribution lies in the second subset. If the true distribution is contained in the third set a noise alarm should occur sooner or later.

Example 3.16. In Example 3.1 we assumed that the das random numbers were iid $B(1,p)$-distributed. Assume that the set of tolerable and non-tolerable distributions are given by the intervals $p \in [0.49, 0.51]$ and $p \in [0.0, 0.47] \cup [0.53, 1.0]$, respectively.

If the algorithmic postprocessing from Example 3.8(a) is applied, $p \in [0.5 - \sqrt{0.02}, 0.5 + \sqrt{0.02}]$ and $p \in [0.0, 0.5 - \sqrt{0.06}] \cup [0.5 + \sqrt{0.06}, 1.0]$ define corresponding sets. For von Neumann's algorithmic postprocessing, these conditions may be even further relaxed. As long as the true distribution of the random numbers is contained in the specified family (here, if the respective random variables remain iid) only performance reasons may enforce a noise alarm. (Note that the output rate shrinks with increasing bias.)

Time intervals or events have to be specified when a basic test shall be executed, e.g., always, one basic test per second, one basic test after each external call for

random numbers, permanent testing within the idle time of the device (if the PTRNG is part of a larger cryptographic system), etc.

If the internal random numbers can be buffered it may be reasonable to apply the following test strategy: If the number of tested internal random numbers in this buffer (i.e., internal random numbers that are ready to be output) falls below a specified bound the buffer is filled up with new internal random numbers. These new internal random numbers or the das random numbers that are generated at the same time form the beginning of a sample to which the online test is applied to. When the online test has been passed also the new random numbers are ready to be output. Otherwise these numbers are deleted.

Alternatively, the online test may be applied *after* the buffer has been filled with new internal numbers. The advantage of this variant is that an adversary has absolutely no information on the stored internal random numbers. We will come back to this topic in Section 3.8 in the light of side-channel attacks and fault attacks.

With regard to the intended applications, a reasonable upper bound for the average number of noise alarms per year should be specified. Assume, for instance, that for a specific smart card ≤ 0.00002 noise alarms per year occur on average if the true distribution of the random numbers is acceptable. If the smart card is set mute after a noise alarm about 20 smart cards per million have to be exchanged unnecessarily per year.

To select appropriate parameters, the designer should be able to compute the probability for a noise alarm within a test suite. Depending on the application and on the applied test strategy ($\rightarrow$ expected number of basic tests per year) this implies the expected number of noise alarms per year. Since each basic test requires many random numbers, it is reasonable to assume that the random variables $C_1, C_2, \ldots$ are iid. Consequently, $(HC_0 = E_{0,r}(C), n_0 = 0), (HC_1, n_1), (HC_2, n_2), \ldots$ forms a homogeneous absorbing Markov chain on the finite state space $\Omega = \{j2^c \mid j2^c \in S_{(C)}\} \times \{0,1,\ldots,k-1\} \cup \{\infty\}$. The number n_j is maximum such that $c_j, c_{j-1}, \ldots, c_{j-n_j+1} \notin S_{(B)}$. The absorbing state ∞ is attained if criterion (B) or (C) is violated. (Recall that criterion (A) is violated only with negligble probability unless the PTRNG has at least (almost) totally broken down.) For details the interested reader is referred to Ref. [40], Section 6.

Example 3.17. The basic test is a χ^2 test on 128 four-bit words while $S_{(A)} = [0.0, 200.0]$, $S_{(B)} = [0.0, 26.75]$ and $S_{(C)} = [13.0, 17.0]$. Further, $\beta = 2^{-6}$, $c = 5$, $k = 3$ and $x = 3$. With regard to the application and the specified test strategy we expect 530,000 basic tests per year. The stochastic model indicates that the random numbers are iid $B(1,p)$-distributed. Table 3.1 collects some numerical results. The term p_{npa} quantifies the probability of a noise pre-alarm within a particular test suite. These figures indicate that for $|0.5 - p| > 0.025$ a noise alarm should occur soon while for $|0.5 - p| \leq 0.01$ noise alarms are relatively rare events. Note, however, that for typical smart card applications smaller noise alarm probabilities for acceptable distributions are necessary, at least if the consequence of a noise alarm is to shut the PTRNG down forever.

Table 3.1 Numerical example.

p	p_{npa}	$E\left(\frac{\#\text{ noise alarms}}{\text{year}}\right)$
0.500	0.0162	0.004
0.495	0.0184	0.006
0.490	0.0289	0.024
0.485	0.0745	0.396
0.480	0.2790	16.6
0.475	0.7470	

Remark 3.9. (see [40], Section 9)
(i) The number N should be a power of 2 to save unnecessary matrix multiplications with large matrices. This minimizes the computation time and reduces round-off errors when computing the probability p_{npa}.
(ii) The smaller $\beta := 2^{-b}$ the smaller is the influence of single basic test values on the history variables $hc_1, hc_2, \ldots$.
(iii) The history variables $HC_0, HC_1, \ldots$ may be interpreted as a 'weighted' random walk on $S_{(C)} \cap \{j2^{-c} \mid j \in Z\}$ with absorbing state ∞. The smaller c the more 'inert' is this random walk and the smaller is p_{npa}.
We recommend to choose $b, c \in \{5,6\}$. (Recall that the transition matrix P has $|\Omega|^2 = (k \cdot |S_{(C)} \cap \{j2^{-c} \mid j \in Z\}| + 1)^2$ entries.).
(iv) Although it is relevant, the meaning of the sample size m of the basic test is often neglected. Consider, for instance, a monobit test with sample size m and $S_{(B)} = [0.5m - \alpha\sqrt{m}, 0.5m + \alpha\sqrt{m}]$ for fixed α. For an ideal RNG, $\text{Prob}(C_j \in S_{(B)}) = \Phi(2\alpha) - \Phi(-2\alpha)$ regardless of m (provided that m is not extremely small). If the random numbers are iid $B(1,p)$-distributed with $p \neq 0.5$ this is yet no longer true. For $p = 0.49$, for instance, the rejection probability is almost the same as for $p = 0.5$ if m is small but almost 1 for very large m (see also Exercise 8).

The next remark addresses important aspects which have not been discussed in this chapter. Remark 3.10(i) refers to fault attacks which will be treated in Section 3.8.

Remark 3.10. (i) Primarily, online and tot tests shall detect unintended weaknesses of the noise source (ageing effects, tolerances of components, total breakdown of the noise source). However, the designer should also consider possible active attacks (fault attacks). Such attacks should either be prevented or detected by physical countermeasures, or at least the distribution of the random numbers should remain in the specified class of distributions ($\rightarrow$ stochastic model), moving to the subset of unacceptable distributions if the quality (entropy) of the random numbers goes down (cf. Section 3.8).
(ii) In this section we only considered eventual misbehavior of the analog part of the PTRNG. However, the algorithmic parts of the RNG might be implemented

incorrectly, or particular components (e.g., an LFSR or a buffer) may become defective Such failures could be detected with known-answer tests (see [20]).

3.7 Alternative Security Philosophies

In this chapter we treated stochastic models and effective online tests. As already pointed out, this shall ensure theoretical security, or more precisely, quantify the average workload to guess random numbers with a non-negligible probability if the adversary has maximum knowhow and unlimited computational power. A reliable stochastic model and effective online tests are mandatory for a successful evaluation with regard to the evaluation guidance AIS 31 ([2, 23]), which has been effective in the German certification scheme (→ Common Criteria, [8, 9]) since 2001. A large number of certification processes have verified the applicability of the AIS 31 to very different PTRNG designs. The Common Criteria themselves do not provide evaluation rules for RNGs. We point out that the AIS 20 and AIS 31 are currently updated. In particular, DRNGs and PTRNGs will be treated in a joint document.

We note that alternative security paradigms exist. The security of a PTRNG may essentially be grounded on a strong cryptographic postprocessing algorithm with memory, so that even a total breakdown of the noise source leaves a DRNG that fulfils requirements (R1) and (R2) or even (R1), (R2) and (R3). In other words, even if the entropy of the das random numbers decreases to 0 the postprocessing still guarantees computational security *provided that the entropy of the memory buffer was maximum at some instant*, which should be the case if the noise source had worked properly for some time. This reduces the requirements on the understanding of the noise source and the effectiveness of the online tests considerably. On the negative side, this does not ensure theoretical security, and (time-consuming) cryptographically strong postprocessing is mandatory. In our understanding, this construction is essentially a hybrid DRNG. The ISO standard 18031 [20] allows both alternatives, namely a (security-proofed) strong noise source with effective online tests, but also a not necessarily strong noise source with a not necessarily effective online test (aka health test) combined with a cryptographically strong postprocessing algorithm.

We note that a combination of both security paradigms, a strong noise source with effective online tests and a (possibly additional) strong cryptographic postprocessing with memory, provides two security anchors, one aiming at theoretical security, the other on practical security. A further advantage of such hybrid PTRNGs is that they may be operated in different modes, depending on the security and functional requirements of the application: as a pure PTRNG (skipping the cryptographic postprocessing), as a hybrid PTRNG (applying the cryptographic postprocessing), or as a pure/hybrid DRNG (without updating the memory of the cryptographic postprocessing algorithm continuously). The third mode might be necessary to achieve high output rates, e.g., to generate blinding or masking values (as a protection against

side-channel attacks) for high-speed encryption. Hybrid RNGs will explicitly be considered in the updated versions of the AIS 20 and AIS 31.

3.8 Side-channel Attacks and Fault Attacks

In the last decade, side-channel attacks and fault attacks have attracted enormous attention in both the scientific community and smart card industry. Unless a cryptographic device is operated in a secure environment side-channel and fault attacks constitute serious threats to any security-relevant operation. Although it is part of the overall security evaluation of the device we briefly address some aspects that concern the PTRNG, comprising the noise source, the algorithmic postprocessing algorithm and the online (tot, self) test.

Of course, the noise source (or more precisely, the das bits) should be resistant against side-channel attacks (in particular against electromagnetic radiation attacks). If the noise source is not properly shielded or sensors do not detect possible fault attacks, the duties of the online test also comprise the detection of non-tolerable weaknesses of the das random numbers (as well as of the internal random numbers) that might be induced by successful fault attacks (cf. Remark 3.10(i)). In [24] an RNG design similar to that in Example 3.6 is discussed with two noisy diodes instead of one, where the difference of the output voltages of both diodes is exploited. The basic idea is to prevent fault attacks since external influences should affect both diodes in the same way. We mention that one has to take care that the output voltages are not too different, which might lower the output rate substantially. The postprocessing algorithm should also be protected. If realized by algorithms from the cryptolibrary, effective solutions should have been developed in connection with the protection of these cryptographic algorithms anyway.

In Section 3.6 we discussed two application schemes for the online test. In the first scheme (here denoted as scheme A) the designated output data themselves are part of a sample to which the online test is applied, while in the second scheme (here denoted as scheme B) the designated output data are buffered first before the online test is applied. Assume the worst case scenario for the moment, namely, that the implementation does not prevent side-channel and/or fault attacks. For scheme A a (maximum) successful side-channel attack might reveal the tested random numbers, and a successful fault attack (applied to the before-buffered random numbers) might fool scheme B. (Note that this fault attack violated the stationarity assumption.) Vice versa, a fault attack on scheme A should cause a noise (pre-)alarm whereas in scheme B a side-channel attack on the online test does not reveal the buffered data.

If this is necessary for the concrete device and the intended conditions of use one might consider a combination of both schemes, hoping that external manipulations cannot be switched on and off at very short intervals. In the easiest case, the das random numbers and the internal random numbers are iid. Then the random numbers may be transmitted alternately over two separate lines, applying scheme A on line 1 and scheme B on line 2. The designated output values are stored in intermediate

buffers. If both online tests have been passed, the content of the intermediate buffers are XOR-ed.

3.9 Exercises

1. Consider Example 3.3 with $m = 2$ (binary-valued random variables) and transition matrix $P = (p_{ij})_{0 \le i,j \le 1}$.
   (i) Determine $H(R_{n+1} \mid R_n)$ if the RNG is in equilibrium state. In particular, compute the numerical values for the special cases $(p_{01} = p_{10} = 0.5)$, $(p_{01} = p_{10} = 0.6)$, $(p_{01} = 0.6,\ p_{10} = 0.4)$ and $(p_{01} = 0.45, p_{10} = 0.51)$.
   (ii) Compute $H(R_{n+1} \mid R_n = i)$ and $H_\infty(R_{n+1} \mid R_n = i)$ for the parameter values from (i) for $i = 0, 1$.
   (iii) Compare and discuss the results from (i) and (ii).
2. Consider a high-frequency oscillator that provides the D-input of a flip-flop. The flip-flop is latched by a low frequency oscillator. Formulate a stochastic model and try to analyse this model. Does this model remain valid if both oscillators are realized as ring oscillators?
3. Improve the lower entropy bound (3.31) in Example 3.9. Hint: Consider the conditional entropy $H(Y_{n+1} \mid Y_n, R_{2n-2})$.
4. Derive a lower entropy bound for Example 3.9 if the von Neumann postprocessing is applied.
5. Consider Example 3.11.
   (a) Evaluate the right-hand integral in (3.34) for $k \in \{3,4,5,6\}$, e.g., by numerical integration or with a computer algebra system. Compare the exact value with the coarse lower bound given in Example 3.11.
   (b) Try to estimate the entropy per internal random number for other distributions than iid $B(1, 0.5)$-distributed das random numbers, e.g., if the das random numbers are $B(1, p)$-distributed with arbitrary p.
6. Propose an effective online test for Example 3.14(ii). Justify your answer.
7. Propose an effective online test for Example 3.14(iii). Justify your answer.
8. Consider Remark 3.9. Determine $\text{Prob}(C_j \in S_{(B)})$ for several parameter values (p, α, m).

3.10 Projects

1. Implement a PTRNG on an FPGA. If you exploit more than one noise source, are these noise sources independent? Try to formulate, justify and analyze a stochastic model.
2. Implement the circuit from Exercise 2 with CMOS chips. Implement both oscillators as ring oscillators. The stochastic model determined in Exercise 2 contains parameters. Use your hardware implementation to determine numerical values for these parameters.

References

1. AIS 20. *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators*. Version 1, 02.12.1999 (mandatory if a German IT security certificate is applied for; English translation). www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf
2. AIS 31. *Functionality Classes and Evaluation Methodology for Physical Random Number Generators*. Version 1, 25.09.2001 (mandatory if a German IT security certificate is applied for; English translation). www.bsi.bund.de/zertifiz/zert/interpr/ais31e.pdf
3. ANSI X9.82. *Random Number Generation* (Draft Version).
4. V. Bagini and M. Bucci. A Design of Reliable True Number Generators for Cryptographic Applications. In Ç. K. Koç and C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 1999*. Springer, Lecture Notes in Computer Science, Vol. 1717, pp. 204–218, Berlin, 1999.
5. M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo. A High-Speed Oscillator-Based Truly Random Number Source for Cryptographic Applications, *IEEE Transactions on Computers*, 52, pp. 403–409, 2003.
6. M. Bucci and R. Lucci. Design of Testable Random Bit Generators. In J. Rao, B. Sunar editors, *Cryptographic Hardware and Embedded Systems—CHES 2005*. Springer, Lecture Notes in Computer Science, Vol. 3659, pp. 147–156 Berlin, 2005.
7. H. Bock, M. Bucci, and R. Luzzi. An Offset-Compensated Oscillator-Based Random Bit Source for Security Applications. In M. Joye, J.-J. Quisquater editors, *Cryptographic Hardware and Embedded Systems—CHES 2004*. Springer, Lecture Notes in Computer Science, Vol. 3156 pp. 268–281, Berlin, 2004.
8. *Common Criteria for Information Technology Security Evaluation*. Part 1–3; Version 3.1, Revision 1, (September 2006) and ISO 15408:1999.
9. *Common Methodology for Information Technology Security Evaluation* CEM-99/045. Part 2: Evaluation Methodology, Version 3.1, Revision 1, September 2006.
10. J.-S. Coron. On the Security of Random Sources. In H. Imai and Y. Zheng editors, *Public Key Cryptography—PKC 99*. Springer, Lecture Notes in Computer Science, Vol. 1560, pp. 29–42, Berlin, 1999.
11. J.-S. Coron and D. Naccache. An Accurate Evaluation of Maurer's Universal Test. In S. Tavares and H. Meijer editors, *Selected Areas in Cryptography—SAC '98*. Springer, Lecture Notes in Computer Science, Vol. 1556, pp. 57–71, Berlin, 1999.
12. L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, 1986.
13. M. Dichtl. How to Predict the Output of a Hardware Random Number Generator. In C. D. Walter, Ç. K. Koç, C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 2003*, Springer, Lecture Notes in Computer Science 2779, pp. 181–188, Berlin, 2003.

14. M. Dichtl and J. Golic. High-Speed True Random Number Generation with Logic Gates Only. In P. Paillier, I. Verbauwhede editors, *Cryptographic Hardware and Embedded Systems—CHES 2007*, Springer, Lecture Notes in Computer Science 4727, pp. 45–62, Berlin, 2007.
15. W. Feller. *An Introduction to Probability Theory and Its Application*, Vol. 1, 4th Revised Printed, Wiley, New York, 1970.
16. W. Feller. *An Introduction to Probability Theory and Its Application*, Vol. 2, Wiley, New York, 1965.
17. W. Hoeffding and H. Robbins. The Central Limit Theorem for Dependent Random Variables. Duke *Mathematical Journal*, 15: 773–780, 1948.
18. N. Howgrave-Graham and N. Smart. Lattice Attacks on Digital Signature Schemes. Des. *Codes Cryptography*, 23: 283–290, 2001.
19. Intel Platform Security Division. *The Intel Random Number Generator*. Intel Corporation, 1999.
20. ISO/IEC 18031. *Random Bit Generation*. November 2005.
21. G. L. Jones. On the Markov Chain Central Limit Theorem. *Probability Surveys*, 1: 299–320, (2004).
22. G. K. Kanji. *100 Statistical Tests*. Sage Publications, London (1995).
23. W. Killmann and W. Schindler. *A Proposal for Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators*. Version 3.1 25.09.2001, mathematical-technical reference of [2] (English translation); www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf
24. W. Killmann and W. Schindler. A Design for a Physical RNG with Ro-bust Entropy Estimators, In E. Oswald and P. Rohatgi editors, *Cryptographic Hardware and Embedded Systems — CHES 2008*. Springer, Lecture Notes in Computer Science, Vol. 5154, pp. 146–163, Berlin, 2008.
25. D. P. Maher and R. J. Rance. Random Number Generators Founded on Signal and Information Theory. In Ç. K. Koç, C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 1999*. Springer, Lecture Notes in Computer Science, Vol. 1717, pp. 219–230, Berlin, 1999.
26. S. Mandal and S. Banerjee. An Integrated CMOS Chaos Generator. In S. Banerjee editor, 1st *Indian National Conference on Nonlinear Systems & Dynamics—NCNSD 2003*. Kharagpur (India), pp. 313–316, 2003.
27. G. Marsaglia. Diehard (Test Suite for Random Number Generators). www.stat.fsu.edu/∼geo/diehard.html
28. U. Maurer. A Universal Statistical Test for Random Bit Generators. *Journal of Cryptology*, 5 1992: 89–105.
29. A. J. Menezes, P. C. v. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1997).
30. J. v. Neumann. Various Techniques for Use in Connection with Random Digits. In A. H. Taub editor, *von Neumann Collected Works*, Vol. 5, Pergamon Press, London, pp. 768–770, 1963.
31. NIST. *Security Requirements for Cryptographic Modules*. FIPS PUB 140-1, 11.04.1994. www.itl.nist.gov/fipspubs/fip140-1.htm

32. NIST. *Security Requirements for Cryptographic Modules*. FIPS PUB 140-2 (25.05.2001) and Change Notice 1, 10.10.2001. csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
33. NIST. *Digital Signature Standard (DSS)*. FIPS PUB 186-2 (27.01.2000) with Change Notice 1, 5.10.2001. csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
34. Y. Peres. Iterating von Neumann's Procedure for Extracting Random Bits. *Annals of Statistics*, 20, 590–597, 1992.
35. J. O. Pliam. *The Disparity Between the Work and the Entropy in Cryptology* 01.02.1999. eprint.iacr.org/complete/
36. J. O. Pliam. Incompatibility of Entropy and Marginal Guesswork in Brute-Force Attacks. In B. K. Roy, E. Okamoto editors, *Indocrypt 2000*, Springer, Lecture Notes in Computer Science, Vol. 2177, Berlin 2000, 67–79.
37. A. Rényi. On the Measure of Entropy and Information. *In Proc. Fourth Berkeley Symp. Math. Stat. Prob. 1* (1960), University of California Press, Berkeley (1961).
38. A. Rukhin et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST Special Publication 800–22 with revisions dated (15.05.2001). csrc.nist.gov/rng/SP800-22b.pdf
39. W. Schindler. *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators*. Version 2.0 02.12.1999, mathematical-technical reference of [1] (English translation); www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf
40. W. Schindler. Efficient Online Tests for True Random Number Generators. In Ç. K. Koç, D. Naccache, C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 2001*. Springer, Lecture Notes in Computer Science, Vol. 2162, pp. 103–117, Berlin, 2001.
41. W. Schindler and W. Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. In B. S. Kaliski Jr., Ç. K. Koç, C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 2002*, Springer, Lecture Notes in Computer Science vol. 2523, pp. 431–449, Berlin, 2003.
42. W. Schindler. A Stochastical Model and Its Analysis for a Physical Random Number Generator Presented at CHES 2002. In K. G. Paterson editor, *Cryptography and Coding—IMA 2003*, Springer, Lecture Notes in Computer Science 2898, pp. 276–289, Berlin, 2003.
43. C. Shannon. Mathematical Theory of Communication. *Bell System Technology*, vol. 27, pp. 379–423, 623–656, 1948.
44. T. Tkacik. A Hardware Random Number Generator. In B. S. Kaliski Jr., Ç. K. Koç, C. Paar editors, *Cryptographic Hardware and Embedded Systems—CHES 2002*, Springer, Lecture Notes in Computer Science, vol. 2523, pp. 450–453, Berlin, 2003.
45. A. M. Yaglom. *Correlation Theory of Stationary and Related Random Functions*, Vol. 1. Springer Series in Statistics, Springer, New York, 1987.