A True Random Number Generator with Built-in Attack Detection

Bernhard Fechner, Andre Osterloh

Department of Computer Science

FernUniversität in Hagen

{Bernhard.Fechner, Andre.Osterloh}@fernuni-hagen.de

Abstract

True random number generators (TRNGs) are extensively used in cryptography, simulations and statistics. Metastability is a way to generate true random numbers. By using electromagnetic radiation, a flip-flop in a metastable state can be manipulated to a known state. In this work, we introduce and analyze the concept of a randomized bit-cell, being able to simultaneously produce random numbers and detect active nonintrusive attacks. The experimental comparison with a standard TRNG yields an 11.5 times better distribution of zeros and ones while the TRNGs are under attack. The concept is extended by using a corrector. A perfect distribution can be gained at the expense of a delay which is proportional to the quality of the random source.

1 Introduction

Random numbers are used and needed in our everyday lives. The generation of random numbers is an essential component to generate passwords, session keys for SSL (secure socket layer), but are also used in simulations such as nuclear physics, weather forecasts, traffic simulations or Monte-Carlo approximations. The quality of a random number generator (RNG) substantially determines the security of the underlying communication. A manipulation leads to predictable random numbers and causes unsafe communication channels. We have to clarify what a *good* random number is. In our case, a good random number is a binary number where zeros and ones are discretely and equally distributed. A discrete random variable X with a finite number of characteristics $\{0,1\}$ is equally distributed, if the probability of each x_i $(i = 1,...,n)$ is equal:

$$P(X = x) = f(x) = \begin{cases} \frac{1}{n} \text{ for } x = x_i (i = 1, ..., n), \\ 0 \text{ else.} \end{cases}$$

2 Metastability

This work deals with true random number generators (TRNG). In contrary to pseudo random-number generators, TRNGs are able to produce a non-periodic sequence of bits. Numerous TRNGs have been developed, some of them with industrial background, e. g. the random number generator within the Intel i810 chipset [2]. TRNGs are based on radioactive decay, ring oscillators [3], thermal noise [2], the photoelectric effect or other quantum phenomena [4]. However, we are aware of the fact, that a TRNG can be subordinated to physical influences. Therefore one should carry out tests like DIEHARD [5] or NIST tests [6] to evaluate the RNG. These tests require a huge amount of random numbers to gain statistical significance and thus a certain time to determine the *actual quality* of the RNG. For fast online tests, others methods have to be regarded.

The violation of setup and hold times within a flip-flop (ff) leads to an unpredictable behavior of the flip-flop called *metastable state*. The probability that the flip-flop will be metastable is an exponentially decreasing function [7]. In [8] an analog-digital converter is used to produce random numbers from the output of a flip-flop being in metastable state although it is unclear on which voltage levels the flip-flop will toggle. For the generation of n-bit random numbers we use the final state of a ff being metastable once.

Three things must be known about metastability [7]:

- What is the likelihood T_0 that the flip-flop will enter a metastable state?
- How long will it be expected to remain in that state (τ)?
- What is the propagation delay h of the flip-flop.

The mean time between failures (MTBF) of a flip-flop in a metastable state results from empirical observations [8] to

$$MTBF(t) = \frac{e^{\frac{t}{\tau}}}{\left(T_0 \cdot f_{clk} \cdot f_{in}\right)},$$

where t is the time a digital output can remain in a metastable state, f_{clk} the system clock frequency and f_{in} the frequency of the input signal. In our fault model, we assume that the attacker is able to change the state of a flip-flop systematically. Therefore, we propose the online injection of a burst of 1 to n stuck-at-0 or stuck-at-1 faults. The observation interval is the time to generate a single random number of length n. In Section 5, we refine the fault model for a worst-case scenario.

3 Randomized Bit Cells

Figure 1 shows the basic randomized bit-cell (RBC). Clock signals have been removed for clarity. All square elements are flip-flops which have random states due to their previous metastable period. The main idea is clear: the flip-flops controlling the decoders are generating random numbers, selecting random elements out of the bit-cell matrix which are also randomized due to their metastable state. The output is a single bit b1. The bits c1-c4 are used for attack detection. If we assume the number of ffs as 2^{2k} arranged in a (2^k, 2^k)-matrix, we have 2k bits for the selection of a single bit cell. This RBC with k bits for the selection is called RBC_k.

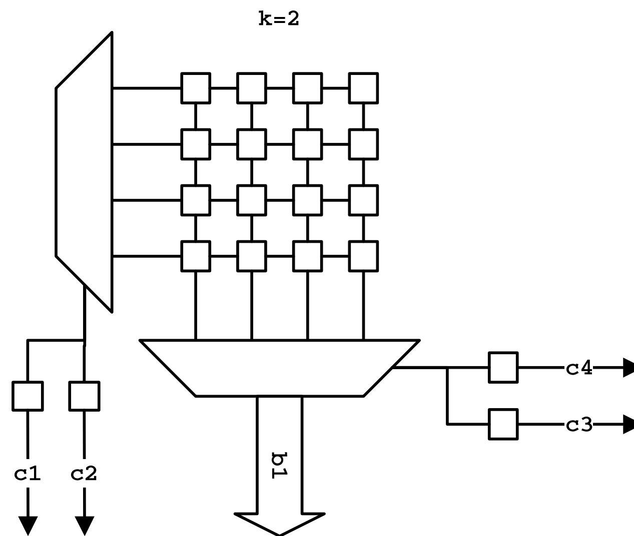

Figure 1: Randomized Bit-Cell (RBC)

The advantage of the scheme in Figure 1 is that the second level (the ffs at the inputs of the decoder) introduces a meta-level of randomization. If an attacker is able to disturb the contents of the decoder ffs, there is still a randomization, because a dedicated bit-cell will be selected, which is again randomized. If the attacker tries to influence the function of a single bit-cell, the randomization to select the bit cell will decrease the probability for a successful attack. This probability computes to $\frac{1}{2^{2k}}$. A single and most common case of a noninvasive attack is left: the influence of all ffs within the RBC. We have 5 random outputs b1 and c1-c4. Electromagnetic radiation will toggle the ffs to a fixed and known state. By comparing the outputs, we can detect these attacks. Figure 2 shows a simple way to detect manipulations. Every output within the array of RBCs is compared locally. If we have a global matching, we signal an attack. A mechanism to detect permanent faults through manipulation of the attack signal is to negate each signal from a RBC to create a second (negated) output. In this case,

every stuck-at fault is detected by comparing both signals by using an external circuit within the perfection core.

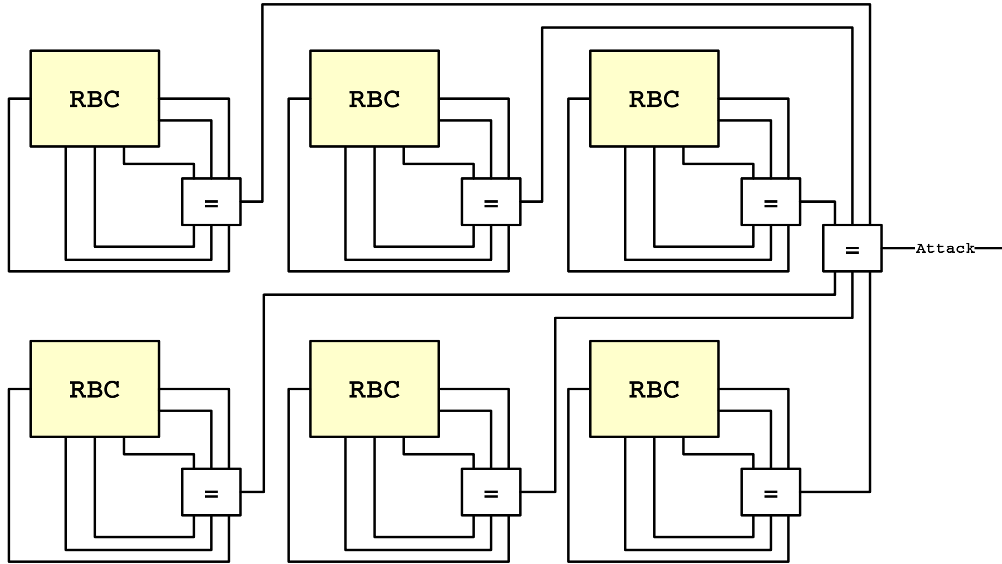

Figure 2: n-Bit RNG with Attack detection

4 Improving the Quality of Randomness with the Smoothing Bit-Cell

In this Section we assume that our random source is not perfect, i.e. the probability that the source produces a zero differs from the probability that it produces a one. Such effects can occur due to the production process or by external circumstances (e. g. a nonintrusive attack). We introduce the concept of a *smoothing bit-cell* (SBC) to reduce such effects and discuss its assets and drawbacks. The *SBC* is a simple extension of a randomized bit cell by using a corrector. In our case, we use a von-Neumann corrector [9]. Let $0<p0, p1<1, p0+p1=1$ the probability that the random source produces a zero or one, respectively. We get the probabilities listed in Table 1 for sequences of length two.

Table 1: Sequences and Probabilities

Sequence	Probability
00	$p0^2$
01	p0p1
10	p1p0
11	$p1^2$

Note that the probability for '01' is the same as for '10'. We exclude the sequences '00' and '11'. To avoid misunderstandings, we call ones and zeros produced by the random source 0- and 1-bits (random bits) and the zeros and ones produced by the proposed method as zeros and ones (random numbers). The drawback is that no random numbers are generated while we have long sequences of 0- or 1-bits. The expected number of bits that we have to wait is

$$\sum_{n=0}^{\infty} 2nq(1-q)^{n-1} = 2/q,$$

where q=2p0p1. If the source produces a 0-bit with probability 2/5, i. e. q=6/25, the expected number of bits we have to wait for is $25/6 \approx 4{,}167$. If we assume a very bad random source producing a 0-bit with probability 10^{-k}, the expected number is approximately 10^{k}. A further advantage is that we are able to detect stuck-at-0 and stuck-at-1 faults or alleviate manipulations of the random source through a timer, measuring the output frequency.

Now we discuss whether this approach can be extended to a sequence of length n>2. The main idea in case of n=2 was that we have to identify sequences with equal probabilities. The only possibility to do this is to take sequences where the number of zeros and ones is equal. Hence we have to restrict our attention to even n. The probability that the sequence exactly contains n/2 zeros is

$$\binom{n}{n/2}(p0)^{\frac{n}{2}}(p1)^{\frac{n}{2}} = (2p0p1)^{\frac{n}{2}}.$$

Thus, we get $q = (2p0p1)^{\frac{n}{2}}$ in this case. Thus, n=2 is the best choice. The number of bits necessary to produce a one or a zero could be seen as a measure for the time needed to produce a random number. To reduce this time we can produce random numbers of k random sources in a synchronized and parallel fashion. The question is how long, i.e. how many steps we have to wait until we get a one or a zero. First, we have to describe how zeros and ones are produced and what will be a *step* in our model. In one step each of the k sources produces one bit. If at least one of k sources separately produces '01' or '10' after two steps, we have an output of zero or one. In case that all k sources produce '00' or '11', we have no output. The probability that there is no output after two steps is $\prod_{i=1}^{k}(1-2p0_i p1_i)$, where $p0_i$ ($p1_i$) is the probability that the random source i produces a zero (one). The expected number of steps we have to wait is

$$\frac{2}{1-\prod_{i=1}^{k}(1-2p0_i p1_i)} = \frac{2}{1-\prod_{i=1}^{k} q_i},$$

where $q_i=1-2p0_ip1_i$. If we assume two random sources with q1=q2=6/25, the expected number of steps we have to wait is approximately 2.122.

5 Probability of Faults

Now we assume that we have one faulty flip-flop and we want to determine the probability that the produced random number is affected by this fault. In our calculation we assume that we have a *worst-case fault*. For our approach with one flip-flop obviously the result is affected with probability one. For the case where we have k ffs in parallel, the probability depends on the way we implement the choice when two or more random numbers are produced by the k random sources. Under the assumption that a random number from k is chosen with probability 1/k and that $p0=p0_i$ and $p1= p1_i$ for all i, we get a probability of

$$\sum_{i=0}^{k-1}\binom{k-1}{i}\frac{1}{i+1}p^i(1-p)^{k-1-i},$$

where p=2p0p1. Table 2 shows the probabilities that a random number is affected by the worst case described above.

Table 2: Worst-Case Results

Number of ffs in matrix	Total number of ffs	Probability	Name
4	6=4+2	1/6	RBC_1
16	20=16+4	1/20	RBC_2
2^{2k}	2^{2k}+2k	1/(2^{2k}+2k)	RBC_k

6 Experimental Results

For a qualitative comparison we conducted fault-injection experiments. Faults were injected according to the fault-model in Section 2. An array of randomized-bit cells, an array of metastable ffs and an array of smoothing bit cells (SBC) were modeled within a software simulation. In the SBC, an additional bit-cell is introduced to hold the previous results. If the current result differs from the last content, we output a random bit according to a fixed coding like (10) $\rightarrow$0 or (01) $\rightarrow$1. Other results are retained. Figure 3 shows the development of distributions for the sequential (SEQ) array, the array of RBC_k's (k=4, MAT) and the array of SBCs (BSEQ) for 1-39 stuck-at-0 bursts for 1000 fault-injection runs. The array holding the structures had the length of 40 bits, since the typical length for session keys ranges from 40-128 bit or 160 bit [1]. The number of zeros and ones was counted for each method. The 1-bit frequencies were divided by the 0-bit frequencies, resulting in the data points in Figure 3.

Additionally, the linear trend for both methods (linear (SEQ), linear (MAT)) is depicted. The trend formulae show that the randomized bit cell gives an 11.5 times better distribution. The SBC had the best distribution (=1).

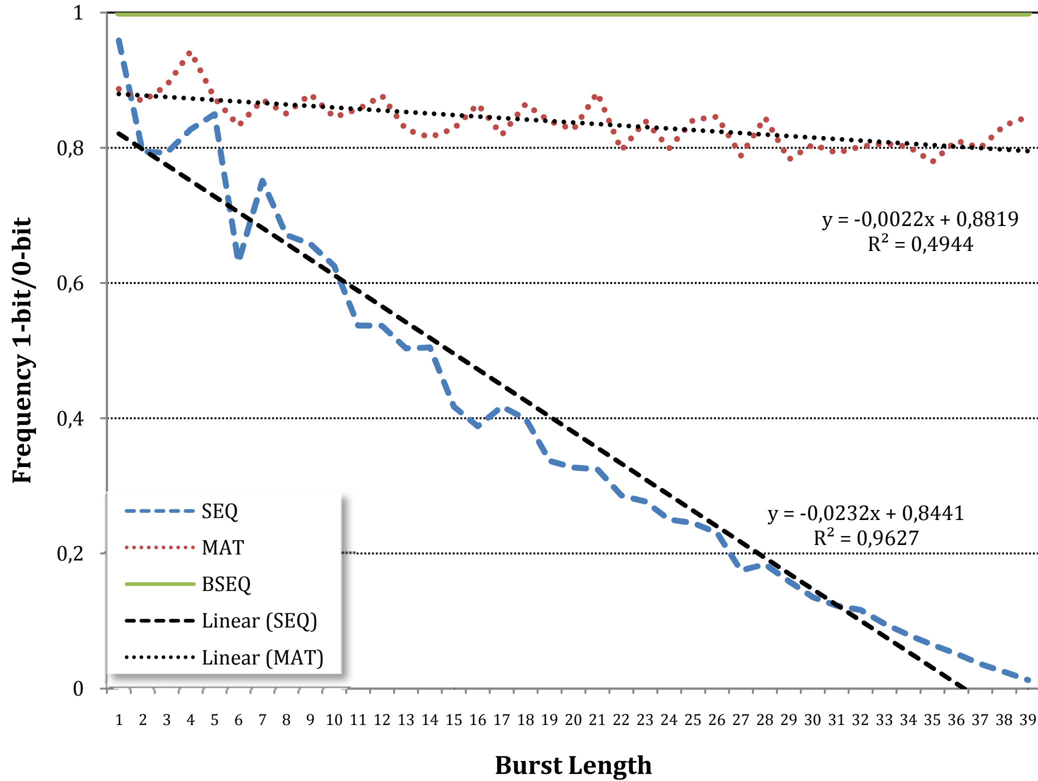

Figure 3: Development of Distributions over 1-39 Burst Faults

7 Conclusion

In this work, we introduced several novelties. First, the concept of the randomized bit-cell (RBC), second the smoothing bit-cell (SBC). Both concepts were experimentally examined and compared with a standard array of metastable memory elements. The experimental results show an 11.5 times better distribution for 1-n stuck-at faults in comparison to the traditional array of metastable elements. Additionally, the SBC was experimentally and theoretically examined. The results show that the SBC gives an equal distribution of zeros and ones to the expense of a delay equal to $1/_{p0p1}$, where p0 and p1 is the probability that a physical random source produces a 0- or 1-bit.

References

[1] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. The Secure Real-time Transport Protocol (SRTP). *Request for Comments 3711.* The Internet Society, 2004.

[2] E. Hoffman. *Random number generator. 5706218* USA, 06. 01 1998.

[3] B. Sunar, W. J. Martin, D. R. Stinson. A Provable Secure True Random Number Generator with Build-In Tolerance to Active Attacks. *IEEE Transactions on Computers.* 2007, Vol. 56, 1.

[4] Quantum Random Bit Generator Service. [Online] 12 12, 2007. [Cited: 12 12, 2007.] http://random.irb.hr/.

[5] G. Marsaglia, DIEHARD: a battery of tests of randomness. *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness.* [Online] 1886. [Cited: 11 22, 2007.] http://www.stat.fsu.edu/pub/diehard/.

[6] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *Special Publication.* 2000, Vol. 800, 22.

[7] Philips Semiconductors. *A metastability primer, Application Note 219.* s.l. : Philips, 1989.

[8] S. Walker, S. Foo. Evaluating Metastability in Electronic Circuits for Random Number Generation. *Workshop on VLSI.* 2001.

[9] B. Kocher, Jun and Paul. *The Intel random number generator.* s.l. : Intel, April 1999.