Self-Timed Ring Based True Random Number Generator: Threat Model and Countermeasures

(Invited Paper)

Grégoire Gimenez*†, Abdelkarim Cherkaoui*, Raphael Frisch*‡, Laurent Fesquet*
*Univ. Grenoble Alpes, CNRS, Grenoble INP**, TIMA, F-38000 Grenoble, France
†Dolphin Integration, F-38240 Meylan, France
‡University of Grenoble, LIG laboratory, F-38330 Montbonnot, France

***Abstract*—Self-timed Ring based True Random Generators (STRNGs) extract randomness from the jitter of events evenly propagating in a Self-Timed Ring (STR) oscillator. Security of such generators is primarily based on an entropy assessment: an accurate model of the minimum entropy per output bit with physical measurement of the noise source. This assessment is reinforced with both entropy source monitoring and online testing of the output bits. This paper addresses the security of the STRNG. First we identify potential vulnerabilities on the generator and define a threat model. Based on this threat model, we analyze the effect of active attacks in analog simulations (in a 55 nm technology), and by emulating them in a high-level simulation model. Then, we propose simple and efficient countermeasures to thwart attacks focusing on the generator. Finally, we evaluate the output sequences before and after attacks to validate the proposed countermeasures.**

I. Introduction

High statistical quality random numbers are required in various cryptographic devices. They serve to generate confidential keys, nonces, random masks and a variety of cryptographic keys. Moreover, they are often used for countermeasures that protect circuits against several types of attacks. These numbers need to be uniformly distributed. Furthermore, in some applications (e.g. symmetric encryption key generation), they also need to be unpredictable.

Uniformly distributed random numbers can easily be obtained with mathematical constructs called Pseudo Random Number Generators (PRNGs). These can be implemented in software or in hardware. Security of such generators is usually based on computational hardness: predicting the algorithm's output is an unsolvable problem. The highest security level is obtained by periodically updating their seed using a True Random Number Generator (TRNG) [1].

TRNGs are mechanical or electrical devices that extract entropy from a physical phenomenon. They generate random numbers which are ideally unpredictable. The concept of unpredictability cannot be measured in the output sequences (e.g. using statistical test batteries). It can only be assessed by measuring the noise source, modeling the extraction mechanism, and estimating Shannon's entropy per output bit [1], [2].

TRNG outputs should be unpredictable. Therefore, they also need to be not manipulable. AIS31 evaluation criteria for TRNGs include internal testing for total failure of the entropy source and some simple online tests to detect major statistical defects of the output sequences [3]. Viktor Fischer further discusses security of TRNGs in [4]. He suggests that even better security can be obtained by directly monitoring the entropy source, e.g. by measuring the noise source even with low precision.

Notice that a compromised RNG will jeopardize the whole cryptographic system security. Thus, it is surprising that very few works adress active attacks on TRNGs. To our knowledge, there are three practical attacks published as of today: power supply manipulation [5], electromagnetic injection attacks [6] and glitch attacks targetting the sampling clock [7].

A reliable TRNG architecture has been proposed in [8] based on a Self-Timed Ring oscillator (STRNG). [9] proposed a simple formula giving a lower bound of the entropy per output bit for STRNGs. Unpredictability (lower bound of entropy close to 1) is guaranteed by setting the appropriate number of ring stages according to measured jitter standard deviation.

In this work, we focus on non-manipulability. We take a closer look at the security of STRNGs by providing ways to guarantee and enforce the unpredictability assessment. Section II describes the architecture of the STRNG and its behavior. Section III defines a threat model for this TRNG and proposes appropriate countermeasures as well as means to monitor the entropy of the output sequences. Section IV evaluates these attacks and the proposed countermeasures using analog simulations in a 55 nm technology as well as a high-level model behavioral model taking into account noise and analog phenomena in the STR. Finally, Section V concludes the paper.

II. Self-timed ring based TRNG (STRNG)

This section presents the STRNG [9] and its mathematical model.

A. Self-timed ring oscillators

1) Architecture and temporal behavior: Self-timed Rings (STR) are oscillators in which events (electrical transitions) propagate without colliding thanks to a handshake request/acknowledgment protocol. The architecture of a STR is depicted in Fig. 1. It corresponds to the control circuit of an asynchronous micropipeline [10], which has been closed to form a ring of L stages. Each stage is composed of a Muller

**Institute of Engineering Univ. Grenoble Alpes

gate with an inverted input R. D_{ff} and D_{rr} are the forward and reverse static propagation delays of a ring stage associated respectively with inputs F and R.

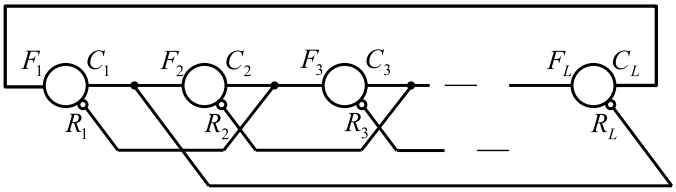

Figure 1: Architecture of a STR

STR stages communicate using a two-phase handshake protocol [10]: unlike in classical ring oscillators, several events can propagate without colliding. The ring is initialized with N events which start propagating in the ring during a transient state. These events end up in a steady state where they arrange themselves in one of two ways, depicted in Fig. 2: either they form a cluster that propagates around the ring (burst oscillation mode), or they spread out all around the ring and propagate with a constant spacing (evenly-spaced oscillation mode) [11].

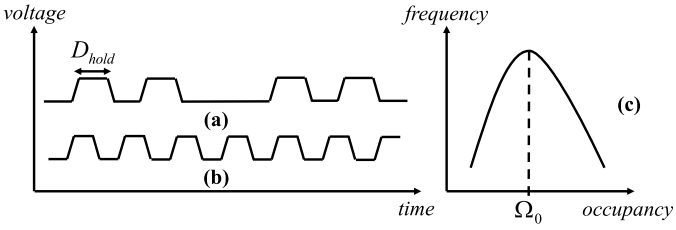

Figure 2: (a) Burst oscillation mode - (b) evenly-spaced oscillation mode - (c) Frequency in the evenly-spaced mode as a function of the ring occupancy Ω

Both oscillation modes are stable and depend on the static parameters of the STR. For a given implementation, they only depend on the ring occupancy $\Omega = N/L$, where L is the number of ring stages and N the number of initialized events. In practice, the evenly-spaced oscillation mode is obtained for an interval of occupancies around Ω_0 given by [12]:

$$\Omega_0 = \frac{D_{ff}}{D_{ff} + D_{rr}} \tag{1}$$

Conversely, the burst mode is obtained for corner values of occupancies.

Moreover, the STR's frequency does not directly depend on its number of stages as in inverter ring oscillators. It rather depends on the ring occupancy Ω. As shown in Fig. 2, it increases with the number of events, it reaches its maximum at Ω_0 defined by Eq. 1, and it starts dropping at higher occupancies.

2) Charlie and drafting effects: STR oscillation modes are due to two analog phenomena in the Muller gate: the Charlie effect and the drafting effect [11]. The Charlie effect represents the impact of the separation time between the inputs events on the gate propagation delay: the closer the events are at the inputs of a Muller gate, the longer is its propagation delay. The drafting effect represents the impact of the time elapsed between successive output commutations on the gate propagation delay: the lower is this elapsed time, the lower is the propagation delay.

In the STR, the Charlie effect causes events to push away from each other when they get close: the propagation delay rises when the separation time at the inputs is lower. On the opposite, the drafting effect causes events to gather together: the propagation delay shrinks as the elapsed time between successive events decreases. When the ring occupancy is low, events gather together and propagate with a hold distance, noted D_{hold} which depends on the Charlie and drafting effects strength. For medium range occupancies, the Charlie effect may become retro-active: events push away from each other until they spread uniformly in the ring. The higher the Charlie effect is, the larger is the interval of occupancies giving the evenly-spaced oscillation mode. Higher occupancies lead to similar behaviors than low occupancies. Events in request paths have to wait for acknowledges: in the asynchronous paradigm, free stages become lower than stages actually processing data. In this case, it is the propagation of acknowledge signals which sets the oscillation frequency. These behaviors are extensively described and modeled in [12], [11] and [13].

3) Sub-gate delay time-stamping using STRs: One major consequence of the above presented features is that STRs can provide uniformly distributed events with a sub-gate time resolution. In classical ring oscillators, this resolution is limited by the propagation delay of one stage: only one event propagates in the ring. Conversely, STRs allow phase differences that are fractions of the propagation delay of one ring stage because several events evolve simultaneously in the ring. Actually, a single event propagation in the ring causes a 90^o phase shift of the oscillating signal. If N events are confined in a L-stage STR and evenly spread around the ring, the phase shift between two stages separated by n stages is [13]:

$$\varphi_n = n \times \frac{N}{L} \times 90^o \tag{2}$$

According to Eq. 2, if the number of stages is a multiple of the number of events, some stages may exhibit the same absolute phase. However, if the number of events and the number of stages are co-prime, the STR exhibits as many different equidistant phases as the number of stages. If T is the oscillation period, the phase resolution in time domain is then:

$$\Delta\varphi = \frac{T}{2L} \tag{3}$$

B. STRNG principle and architecture

The STRNG architecture and principle chronogram are depicted in Fig. 3. The STR is set in its evenly-spaced propagation mode, with a number of events N co-prime with its number of stages L. This ring provides L signals $(C_i)_{1\leq i\leq L}$ having the same period T, a constant phase difference $\Delta\varphi$ between them, and distributed over half an oscillation period of the STR output ($L\Delta\varphi = \frac{T}{2}$). These signals are subject to

jitter variations. For each event, they are represented in Fig. 3 with shaded rectangles and Gaussians around its mean time. In Fig. 3, signals are re-indexed by order of their events arrival time.

Each signal C_i is sampled with the same reference clock clk using a flip-flop. Therefore, whatever the sampling moment t, there exist j such as $|t - t_j| \leq \frac{\Delta\varphi}{2}$, where t_j is the switching moment of the signal C_j. If jitter variations are larger than the phase difference $\Delta\varphi$, the signal C_j is sampled in its uncertainty zone as shown in Fig. 3. The obtained sample is then random, and subsequently the output of the XOR tree also.

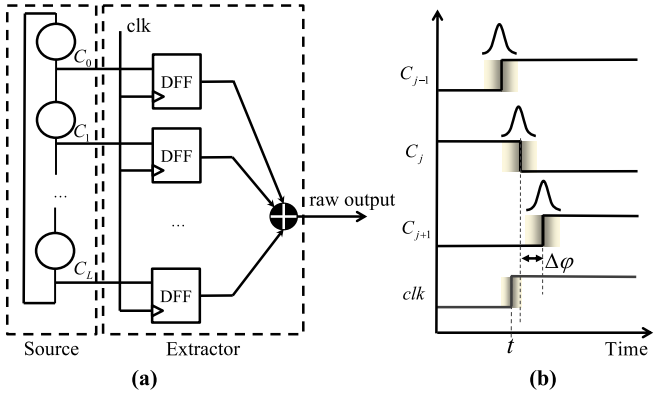

Figure 3: (a) STRNG Architecture (b) Principle chronogram

The main feature of this design is its ability to precisely adjust the relative phases of successive events to the magnitude of the jitter. By increasing L while keeping the same ratio N/L, the ring frequency remains constant. This way, mean value of $\Delta\varphi$ can theoretically be set as small as needed.

C. Entropy model

[9] proposes a stochastic model for the STRNG, allowing to compute a lower bound of entropy per output bit H_m as a function of the jitter magnitude and the STR parameters. H_m is a function of the jitter magnitude σ (the measured standard deviation of the propagation delay of one ring stage), T the STR oscillation period, and L the number of ring stages. P_0 is the probability of sampling a '0' in the worst case, i.e. when the sampling occurs the furthest away from STR edges. H_m is computed using the following set of equations:

$$H_m = -P_0 \log_2(P_0) - (1 - P_0) \log_2(1 - P_0) \quad (4)$$

$$P_0 = 1 - 2\phi(\frac{T}{4L\sigma}) + 2(\phi(\frac{T}{4L\sigma}))^2 \quad (5)$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-t^2}{2}} dt \quad , \; x \in \mathbb{R} \quad (6)$$

By setting this minimum bound near 1 (by selecting the appropriate number of ring stages L), one can garantee that the sampled bit is unpredictable whatever the sampling moment.

III. Threat model and proposed countermeasures

This section presents an extended threat model of STRNGs. We propose a general model that focuses on threats targeting both the entropy extractor and the entropy source. Moreover, a set of countermeasures to protect the STRNG is presented. In this paper, we focus on active attacks able to control the output of the STRNG by modifying its behavior. Passive attacks, which try to anticipate the random values, are powerful attacks but are out of scope of this paper. Furthermore, resilience of a TRNG against passive attacks is guaranteed once unpredictability of its output sequences is proven.

A. Attacks on STRNG: related work

To the best of our knowledge, only one paper has been published on the security of the STRNG. In [7], H. Martín *et. al.* have stressed the generator with three types of attacks, and analyzed their impact on the output sequences. Firstly, they applied environmental manipulations with underpowering and high-temperature. Both attacks showed a limited impact on the output sequences. Increasing temperature reduced the ring frequency, and hence, increased its phase resolution. However, it also increased thermal noise. Since the two phenomena compensated each other, the quality of the output sequences was not significantly impacted. On the other hand, the ring frequency decreased with the power supply voltage. Although the impact on the phase resolution is more perceptible than for temperature, the subsequent entropy loss is easily compensated with arithmetic post-processing consisting of a parity filter [7].

Furthermore, H. Martín *et. al.* present attacks using power supply glitches or clock glitches that successfully produce bias in the output sequences. Power glitches temporarily shut down the system, and may affect the ring behavior. Overclocking produces timing violations in the XOR-tree. Authors show that it is possible to control the output of the STRNG by compromising the XOR-tree. Moreover, they propose elementary countermeasures that protect against this kind of attack. They show that a more pipelined version of the extractor (they implement a ripple structure of the XOR-tree) relaxes the critical-path and reduces this threat. They also presents a lightweight countermeasure based on a filter structure that protects the STRNG against clock glitches.

B. Threat model

In this work, we have identified six threats which can be separated into threats on the entropy source and threats on the entropy extractor. Threats on the STR can be classified in two categories:

1) Threats which modify the ring frequency without compromising the uniform distribution of events (degraded performances)

2) Threats which potentially compromise the uniform distribution of events (incorrect behavior)

Fig. 4 classifies those threats depending on their target (entropy source or extractor) and on their effect on the TRNG behavior.

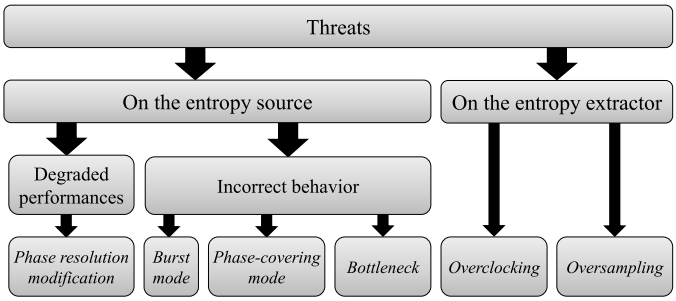

Figure 4: Classification of threats on STRNGs

1) Overclocking: By reducing the clock period (even for a very short time), one can generate timing violations and hence incorrect behavior of the circuit. The STR falls in the Delay Insensitive (DI) class of asynchronous circuits. It is thus not responsive to overclocking. However, the extractor can be impacted by such attacks and thus give the control of the output bits to the aggressor. Notice that increasing the datapath propagation time while keeping the same clock period will generate similar timing violations. Thus the proposed overclocking threat also includes such attacks.

2) Oversampling: It consists in forcing the extractor to sample inter-dependent values. This is achieved by changing the ratio between the STR frequency and the system clock frequency. The inserted redundancy reduces the entropy of each generated random word and thus compromises the security of any cryptographic application using it.

3) Phase resolution modification: H. Martín et al. have demonstrated that STRNGs are robust against environment variations (temperature, voltage) [7]. The STR frequency is actually more stable than classic inverter ring oscillators [14]. Slight over-design handles such variations and allows to keep a very good entropy over the full range of operation. But it is highly probable that powerful active attacks such as EMAs (ElectroMagnetic Attacks) or OFIAs (Optical Fault Induction Attacks) can modify the frequency of the STR while keeping the equidistant phases in the ring. According to equation 3, it will result in an increase of the phase resolution that indeed decreases entropy.

4) Burst-mode: Another relevant threat is to force the STR into its burst oscillation mode. In this case, the TRNG principle is compromised. The lower bound of entropy per output bit is drastically reduced, and the output sequences become predictable. Modifying the number of events propagating in the STR is a plausible attack, any FIA (Fault Injection Attack) and particularly reset-glitch attacks should be considered.

5) Phase-covering-mode: Phase-covering-mode is another insecure state of the STR. This state is observable when the number of events and the number of stages are not co-prime. This means there exist a common divisor X between N and L. In this case, events are gathered in groups of X phases according to Eq. 2. The resulting $\Delta\varphi$ is then multiplied by the same factor X and the minimum entropy is highly impacted.

6) Bottleneck: If one ring stage has a propagation delay several times larger than the others, it acts as a bottleneck for the events propagation. The separation time of this stage inputs is so large than the Charlie effect is negligible. Events gather at the input of this stage, creating a queue where the propagation of the acknowledge signals limits the events flow. Conversely, right after the incriminated stages, events propagate freely until they reach the last event of the queue. When the ring exhibits a bottleneck, its oscillation period depends only on this long stage delay. This behavior is similar to burst-mode, but it may be confused with a normal evenly-spaced mode as it produces signals with 50% cycle times. However, in this case, the correct behavior of the generator is not guaranteed anymore.

C. Active attacks on the entropy source

In this section, we present two active attacks that target the entropy source. These attacks are efficient since they can potentially invalidate the TRNG behavior. Each of them can activate one or several of the threats discussed in the previous section. Furthermore, they are easily implementable at reasonable cost.

1) Token/bubble injection: The token and bubble concept is often used to describe the state of asynchronous controllers. A token means that the stage is processing data, while a bubble signifies that the stage is free and ready to process new data. In the STR, a stage contains a token (T) if its output is different from its input. It contains a bubble (B) if its output is equal to its input. With this formalism and the stage truth table, it can be noted that tokens propagate to next stages if and only if the next stage contains a bubble. Tokens are interpreted as events which propagate in the ring.

One attack consists in modifying the number of events propagating in the ring. When only few events are added/removed, the STR may still exhibit evenly-spaced oscillation mode, but its phase resolution can be slightly impacted. However, if the attack puts the ring into a burst mode or phase-covering mode, the quality of the computed random bit sequence will be highly degraded. Depending on the STR size and configuration, many events must potentially be inserted/removed to provoke the burst-mode. However, very small alteration of the number of tokens may be sufficient to provoke phase-covering mode if the ratio N/L is favorable. For example, according to Eq. 2, removing two tokens from a 125-stage STR with 62 tokens increases $\Delta\varphi$ by a factor 5 (5 is the greatest common divisor of 125 and 60).

For a given ring and at any moment, reading the value held in each stage provides the localization of tokens and bubbles. Two different adjacent values should be interpreted as a token (T) while two identical adjacent values represent a bubble (B). For instance, the pattern ”001110101” must be interpreted as ”BTBBTTTTT”. It represents the state of a 9-stages STR with 6 events propagating in the request paths.

Notice that, due to the ring topology, only even numbers of tokens can be initialized in a self-timed ring. This rule also applies to the addition and removal of tokens. Thus, removing

two tokens from the 9-stage ring to get a "BTBBTBBTT" pattern is simply equivalent to force the value of the seventh bit from 1 to 0 in order to obtain the pattern "001110001". More generally, removing two tokens from a ring can be done by changing any "0[1]*0" pattern into a "0[0]*0" pattern, or by changing any pattern "1[0]*1" to "1[1]*1", where [1]* refers to an arbitrary number of successive '1' values.

There exist different techniques to perform such attack. If the STR initialization vector (which defines set and $reset$ signals of each stage) is accessible to the aggressor, he can easily change the ring configuration. Furthermore, applying a glitch on initialization signals may remove or add tokens. Fault injection attacks should give reproducible results if the attacker is able to target specific parts of the STR and to force the state of each stage individually. Laser can be potentially used to inject such faults in the ring. However, a less invasive attack using electromagnetic pulse might be efficient too and less expensive.

2) Delay modification: The second attack aims at adding delay in one or several stages of the STR. This attack does not change the number of tokens in ring, but it decreases the oscillation frequency. If the attacker adds a slight delay, the STR may keep an equidistant phase distribution mode. In this case, the frequency drop leads to an increase of the phase resolution and, thus, to a lower entropy. However, such modification may not be sufficient to effectively degrade the statistical quality of the output sequences, especially if post-processing is used (usually it is). In fact, most generated bits yield more entropy than the estimated lower bound.

The attacker can drastically increase a stage delay. At some point, he can create a bottleneck. In this case, correct behavior of the generator cannot be guaranteed and the lower bound of entropy per output bit decreases drastically.

Several techniques can be potentially exploited to insert such a delay in a CMOS gate. Environmental manipulations (voltage, temperature) are the simplest, but they have a global effect: they impact both the frequency and the jitter, but do not compromise the evenly-spaced propagation of events. Electronmagnetic attacks that modify the delays in a design should also be considered.

D. Proposed countermeasures and implementation

This section presents several countermeasures to prevent or detect most of the presented attacks.

1) Carefully selecting the number of stages: A basic countermeasure can protect the STRNG against attacks trying to provoke a phase-covering mode. It consists of choosing, during design phase, a number of stages L for which $\forall N \in \mathbb{N} \cap [2, L-2]$, the ratio L/N cannot be reduced. This can be simply done by selecting L as a prime number. In this case, the ring is protected against any token/bubble modifications as long as the STR stays in the evenly-spaced mode. However, this countermeasure does not protect against alterations of the ring structure. Indeed, it is worth considering that, using a Focus Ion Beam (FIB), an attacker can directly modify the ring structure and remove some stages. An aggressor can then modify the ratio between the number of stages and the number of events (L/N). However, such attacks are very complex and expensive. They are also thwarted by any invasive-attack detectors.

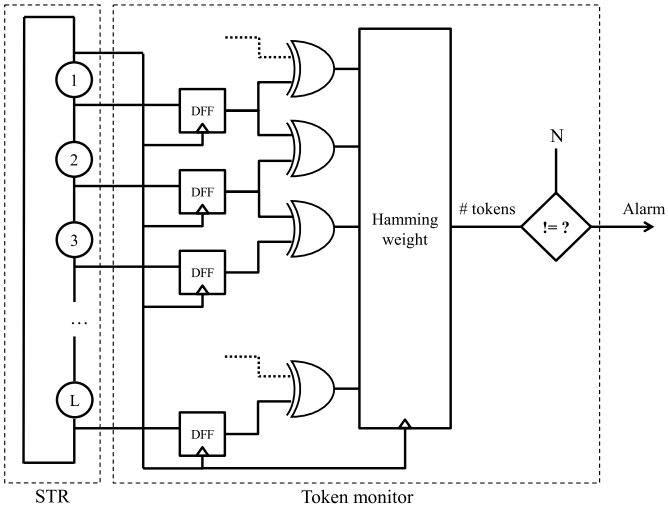

Figure 5: Architecture of the proposed token monitor

2) Event counter: Changing the number of events in the STR also affects the lower bound of entropy if the phase resolution is sufficiently increased. However, in normal operation (without attacks), the number of tokens circulating in the STR cannot fluctuate (it is set at initialization). Based on this fact, a very simple countermeasure consists in monitoring the number of events in the ring. Such a monitor raises an alarm when a modification is detected, which can be interpreted as an intrusion attempt. Appropriate reaction, like TRNG re-initialization, should be performed. Figure 5 shows the architecture of the proposed monitor. The input and the output of each individual Muller gate are XORed together. According to the token/bubble abstraction model, an event is processed by the stage if the two signals are different and hence the XOR gate outputs a '1'. The number of events is then extracted by measuring the Hamming weight of the vector composed by the L XOR-outputs.

3) Internal mode for easy monitoring: AIS31 requires embedded tests to monitor the quality of the random numbers generated [1]. However, these tests do not detect pseudo-randomness. They only assess that the output bits are uniformly distributed. In other words, a failure of the entropy source (i.e. violation of the stochastic model) may not be detected if the pseudo-randomness generated through the sampling of the STR phases with an external clock is still sufficient to validate the online tests. To treat this limitation, we propose to use the delayed output of one signal of the ring as sampling clock, also called internal mode. In this case, the sampling clock is synchronized with the STR output and no pseudo-randomness is generated because there is no deterministic phase drift between them. In this situation, simple online tests as proposed in [3] are more reliable: it is ruled out that they

pass due to pseudo-randomness, and if they fail, it is probable that the entropy per output bit is not sufficient.

IV. Evaluation results

In this work, both token injection attacks and delay modification attacks have been evaluated using mixed-signal simulations with UMC (United Microelectronics Corp.) CMOS 55nm technology models. In addition, a high-level model has been developped for analysis purposes. It includes analog effects of the STR by modeling the Charlie and drafting effects following the timing model of the Muller gate discussed in [12] and [11]. It also models jitter at the level of each logic gate by varying its propagation delay during execution following a normal distribution with parametrable mean value and standard deviation. This way, a high-level model of the TRNG can be impelemented and its output bitstreams can be extracted and evaluated. Attacks have been emulated with this behavioral model and results of statistical tests assess the validity of the proposed threat model and the efficiency of the proposed countermeasures. In the sequel, we present the results of our experiments.

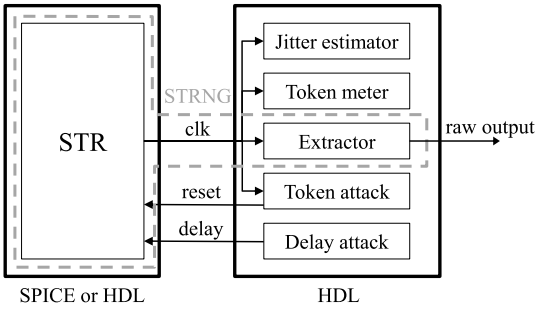

Figure 6: Simulation set up

A. Analog simulations

A mixed-signal environment based on Dolphin Integration's SMASH tool has been used for this analysis. Entropy source is modeled in SPICE using CMOS 55 nm transistors model while the entropy extractor countermeasures are described in HDL. Thus, the attack sequences can be easily built at RTL level and their precise impact evaluated in analog simulations. Fig. 6 shows the structure of the simulation environment.

For the needs of the demonstration, a STR with 125 stages has been built and initialized with 62 tokens. The token removal attacks have been performed using the set signals of the four last stages (121 to 124) as soon as a "1001" pattern is detected. Fig. 7 presents a trace of the attack and its impact on the STR phases. During the attack one can see the token counter decreasing from 62 to 60. This proves the feasibility of the attack. The attack window is only 80 ps large in our case. However, it should be larger once the permanent mode of the STR is reached and when all RC parasitics are included in the simulation. Reducing power supply voltage and increasing temperature will furthermore enlarge the attack window. However, it should be technically very difficult and

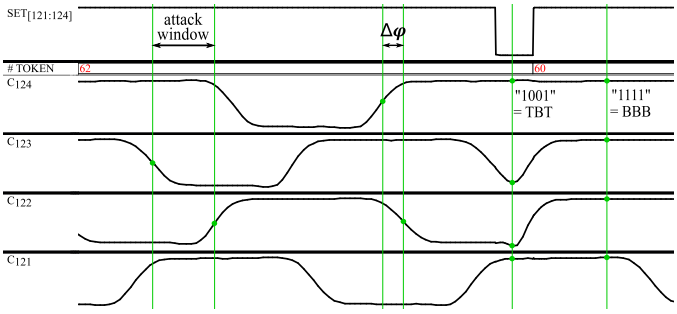

Figure 7: Analog simulation of token injection attack

expensive for an attacker to simultaneously monitor four stages and synchronize a set or $reset$ glitch during such a narrow window. We apply a glitch of 50 ps but there is no restriction on its size to effectively implement the attack. A larger pulse would temporary stop every incoming token. The STR will come back to its standard propagation mode once the $reset$ is relaxed.

Less precise attacks with no synchronization of the $reset$ glitches can be considered. However, it appears that the STR is quite resilient to such attacks. Indeed, in the evenly-spaced mode, tokens are uniformly spread over the ring. It means that, when many events are present in the self-timed ring, the pattern "010" is more probable. On the other hand, when few tokens propagate, the pattern "0[1]*0" is more frequent. Furthermore, when the ratio $\Omega = N/L$ tends to 1/2, events are distributed with a periodic "[0011]*" pattern. An arbitrary $reset$ glitch will not have the same effect depending on the occupancy ratio Ω. This glitch tends to remove tokens when there are many tokens (change "010" into "000") and to add tokens when there is a majority of bubbles (change "111" to "101"). Nonetheless, we remark that such non-controlled reset-glitch attacks have a high probability of leaving the ring configuration unchanged.

We also implemented delay attacks. A delay cell, usually used to fix hold timing in digital circuit, is inserted on the request signal between two stages of the STR. Other techniques may be used, like additional capacitance or directly modifying the SPICE model of one Muller gate. However, the delay cell seems to be the most predictive way to do it, particularly because characterization figures were available through the liberty model of this standard cells. We performed two attacks, one with a delay value similar to the propagating time of a Muller gate and a second with a delay approximately 5 times bigger. As expected, the first attack does not modify the oscillating mode of the STR: even if the frequency is slightly reduced, the ring keeps its evenly-spaced distribution. However, the attack using a bigger delay successfully provokes a bottleneck. In this mode, a part of the tokens are queuing before the stage where the delay has been inserted and the propagation of the acknowledge signal limits the token flow. The other part of the ring, starting from the second stage after the delay, exhibits a fast propagation of tokens, which freely propagate until they reach the last event of the queue.

Threat class	Main effect	Attack	T_{str} (ns)	H_m	FIPS 140-1	Event counter
None (reference)	N/A	N/A	2.130	0.99	10/10	ok
Degraded performances	$\Delta\varphi$ modification	Remove 10 tokens	2.394	0.98	10/10	alarm
	$\Delta\varphi$ modification	Add 100ps delay	2.414	0.98	10/10	ok
Incorrect behavior	Phase covering mode	Remove 2 tokens	2.160	0.19	10/10	alarm
	Phase covering mode	Remove 12 tokens	2.544	~ 0	0/10	alarm
	Burst mode	Remove 52 tokens	12.522	~ 0	2/10	alarm
	Bottleneck mode	Add 1ns delay	5.993	~ 0	0/10	ok

Table I: Attacks and their effects on the STRNG: STR oscillation period T_{STR}, lower entropy bound per output bit H_m, number of sequences that pass FIPS 140-1 out of 10 sequences, output of the token monitor alarm

B. High-level simulations

Starting from the same environment, spice netlist of the STR can be replaced by a high-level behavioral model. Using less complex model shortens the simulation runtimes and allows to generate the random bits sequences required to perform statistical tests. The high-level model is written in HDL and integrates the Charlie and drafting effects of the Muller gate. This model is then partially timed to add the analog behavior of each elementary stages of the STR. The jitter is also modeled at the level of each logic gate. The mean value of the propagation delay of one stage (excluding the Charlie and drafting effect) is of 500 ps and jitter standard deviation has been set to 10 ps for each ring stage.

The testbench of the analog simulations is kept and we performed the same token injection and delay modification attacks than in analog simulation. For each attack, we measure the oscillation frequency and we compute the lower entropy bound -using Eqs. 2, 4, 5, and 6. Then, we extract 20 sequence of 20000 bits that we evaluate using FIPS 140-1 statistical test suite.

C. Results

Results of the previous experiments are summerized in Table I. Table I describes, for each attack: the expected effect, the measured oscillation frequency T_{STR}, the computed lower entropy bound H_m, the number of sequences that passed FIPS 140-1 out of 10 sequences, and finally the token monitor output state (alarm active or not).

The number of ring stages in the reference design is set up according the jitter standard deviation parameter in order to provide a lower entropy bound H_m near 0.99. When no attack is applied, the ring frequency is of 2.13 ns. The token monitor output alarm is inactive. As expected from this setup, all of the output sequences pass FIPS tests.

1) Delay modification attacks: When we add 100 ps delay ($1/5$ of the mean propagation delay of one stage), the ring is not affected except during the attack: events quickly re-converge to a steady evenly-spaced propagation regime. However, as expected, lower entropy bound is barely impacted although the ring frequency is slightly modified. The output sequences are still uniformly distributed (they pass the FIPS tests).

However, if we add 1 ns delay (2 times the mean propagation delay of one stage), then the ring exhibits a bottleneck. Simulation shows that events are not uniformly distributed in the ring: the ring stage with a long delay has a large phase difference with the nearest other signal phases. As a consequence, the lower entropy bound (which is computed using the worst case phase difference), approaches 0. Subsequently, none of the output sequences passed FIPS tests.

Note that the token monitor does not detect such attacks. However, we can observe that in this attack, the ring frequency drastically increased. A countermeasure against such threats, based on a monitor of the ring frequency, is a promising solution that will be studied is future works.

2) Token injection attacks: If we remove 10 tokens from our initial configuration, the ring still exhibits an evenly-spaced propagation of 52 events in 125 stages. Note that 52 and 125 are still co-prime. Thus, as expected, lower entropy bound is barely impacted although the ring frequency is slightly modified. As a result, the output sequences pass the FIPS tests. In addition, the token monitor detected that the number of events has been modified.

If we remove 2 or 10 tokens from our initial configuration, the ring still exhibits an evenly-spaced propagation but is in phase-covering mode. The number of events is not co-prime with the number of stages anymore. In this case, lower entropy bound is significantly impacted and the output sequences may not pass the statistical tests. Note however that the actual value of entropy per output bit can be way higher than the estimated lower bound, but this cannot be guaranteed by the designer (he does not control the phase between the sampling signal and the STR outputs). For instance, the configuration in which two tokens have been removed still passed all the tests because the sampling moment was synchronized with one signal edge with a very low phase difference. Nonetheless, the token monitor detected that the number of events has been modified.

Finally, if we remove 52 tokens from the ring, it enters its burst oscillation mode. In this case, the lower entropy bound is near 0 and the output sequences do not pass FIPS tests, as expected.

3) Alarm management: Since the token monitor counts the number of events in the STR, it can be configured to provide alarms with different severities. Be N_m the new number of events after an attack modifying it. If N_m is not co-prime with L, and if N_m is outside of the occupancies values giving the evenly-spaced mode, then a total failure alarm (TF alarm) must be produced. In this, case the TRNG must be re-initialized. If N_m is such that only the ring frequency is modified, a low-level alarm must be produced. Depending on the TRNG context of usage, re-initialization may not be necessary. For example, if cryptographic post-processing is used (basically a PRNG at the TRNG ouput), then the output bits may still be used as long as the alarm is not active longer than the PRNG period. If a low-level alarm is active for too long (with regards to the output PRNG periodicity), it should be transformed into a TF alarm.

V. Conclusion

Security of TRNGs is based on both the unpredictability and the non-manipulability of their output. STRNGs are a new class of TRNGs based on asynchronous design techniques. The unpredictability of their output is assessed using a stochastic model and physical measurements of the noise source. In this work, we analyzed the security of STRNGs and provided ways to monitor their outputs and protect them against attacks. In particular, we identified two attacks that may severly threaten the STRNG: token injection attacks and delay modification attacks. We analyzed their effect by emulating them in analog simulations and by testing the compromised output sequences. We showed that these attacks can drastically reduce the entropy of the output bits (even near 0) and produce output sequences with major statistical defects. Finally, we proposed three countermeasures at architectural level to thwart these attacks: selecting a prime number of stages, using an internal STR output as a sampling clock, and monitoring the number of events in the ring. By following these simple design rules, and by monitoring the entropy source - in addition to traditional on-line tests -, the designer can guarantee at any time that the output is not under manipulation.

Acknowledgment

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) and by Dolphin Integration.

References

[1] W. Schindler and W. Killmann, "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications," in *Cryptographic Hardware and Embedded Systems*, vol. 2523, 2003, pp. 431–449. [Online]. Available: http://www.springerlink.com/index/XXLBPHW5WYFP923Q.pdf

[2] W. Killmann and W. Schindler, "A design for a physical RNG with robust entropy estimators," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5154 LNCS, 2008, pp. 146–163.

[3] W. Schindler and W. Killmann, "AIS 31: Functionality classes and evaluation methodology for true physical) random number generators, version 2.0," *Bundesamt fuer Sicherheit in der Informationstechnik (BSI), Bonn*, 2011.

[4] V. Fischer, "A closer look at security in random number generators design," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7275 LNCS, 2012, pp. 167–182.

[5] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5747 LNCS, 2009, pp. 317–331.

[6] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, and P. Maurine, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7275 LNCS, 2012, pp. 151–166.

[7] H. Martin, T. Korak, E. S. Millan, and M. Hutter, "Fault attacks on STRNGs: Impact of glitches, temperature, and underpowering on randomness," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 266–277, 2015.

[8] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "A self-timed ring based true random number generator," in *Proceedings - International Symposium on Asynchronous Circuits and Systems*, 2013, pp. 99–106.

[9] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A very high speed true random number generator with entropy assessment," in *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 179–196.

[10] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989. [Online]. Available: http://doi.acm.org/10.1145/63526.63532

[11] A. Winstanley and M. R. Greenstreet, "Temporal properties of self-timed rings," *Tiziana Margaria and Thomas F. Melham, editors, CHARME, volume 2144 of Lecture Notes in Computer Science*, 2001.

[12] J. Hamon, L. Fesquet, B. Miscopein, and M. Renaudin, "High-level time-accurate model for the design of self-timed ring oscillators," in *Proceedings - International Symposium on Asynchronous Circuits and Systems*, 2008, pp. 29–38.

[13] S. Fairbanks, "High Precision Timing using Self-timed Circuits," University of Cambridge, Tech. Rep., 2009.

[14] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet, "Comparison of Self-Timed Ring and Inverter Ring Oscillators as entropy sources in FPGAs," *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1325–1330, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6176697