Non-Deterministic Timers for Hardware Trojan Activation (Or How a Little Randomness Can Go the *Wrong* Way)

Frank Imeson, Saeed Nejati, Siddharth Garg, and Mahesh Tripunitara [*]

Abstract

The security of digital Integrated Circuits (ICs) is essential to the security of a computer system that comprises them. A particularly pernicious attack is the insertion of a hardware backdoor, that is triggered in the field using a timer that is also inserted in the hardware. Prior work has addressed deterministic timer-based triggers — those that are designed to trigger at a specific time with probability 1. We address open questions related to the feasibility of realizing non-deterministic timer-based triggers in hardware — those that are designed with a random component. We show that such timers can be realized in hardware in a manner that is impractical to detect or disable using existing countermeasures of which are aware. We discuss our design, implementation and analysis of such a timer. We show that the attacker can have surprisingly fine-grained control over the time-window within which the timer triggers. Our timer has several other appealing features as well, from the attacker's standpoint. For example, it is practical and effective with only a few bits of Non-Volatile (NV) memory and a small time-window within which volatile state needs to be maintained. Our work raises the bar considerably for defense mechanisms for hardware security.

1 Introduction

The insertion of malicious backdoors, also called hardware Trojans, in the design or manufacturing process of an integrated circuit (IC) is a critical emerging security threat. While the dangerous effects of compromise of software are well-recognized, the compromise of hardware can be even more harmful for two reasons [10]. First, a hardware attack is more persistent, given that hardware is not as easy to patch as software. Second, hardware is at the lowest level of the computer system, a vulnerability in hardware affects the software that runs above it as well. A hardware Trojan can compromise the security of an IC in several ways [22]. For instance, it can allow an attacker to surreptitiously leak or modify sensitive information on the chip, maliciously modify its functionality, or effect unauthorized privilege escalation. Hardware Trojans are usually passive until they are *triggered*. The trigger is a critical component of a hardware Trojan attack. In particular, it allows the hardware Trojan to escape detection during post-fabrication IC testing and instead allows the attack to be activated at a time of the attacker's choosing.

Triggers to activate hardware Trojans are one of two types [24]: *data-* and *time-based*. With the former, an attack is initiated by a cheat-code — a sequence of bits embedded in the data or instruction stream. With the latter, an attack takes place after the elapse of some time, measured, for example, as CPU cycles. While both data- and time-triggered attacks can be destructive, time-based triggers seem to be particularly dangerous because they do not require any collusion at run-time from a higher-level component such as software or a user.

A natural implementation of a time-based trigger is a finite state machine (FSM) that implements an up-counter and triggers the attack when the count reaches a certain value. The state of the counter can be stored on-chip in either volatile registers, in on-chip non-volatile (NV) memory or a combination of the two. We refer to such implementations as *deterministic* time-based triggers (or **D-timers**) as they trigger with probability 1 after a number of clock cycles chosen by the attacker.

Waksman and Sethumadhavan [25] observe that D-timers that use volatile state only are easily defeated by power-cycling the chip. They then propose an improved D-timer that copies its state to on-chip NV memory before power cycling events. However, they observe that this improved attack can also be thwarted. First, the chip

[*]S. Garg is with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, USA. F. Imeson, S. Nejati, and M. Tripunitara are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Contact e-mail: sg175@nyu.edu

is repeatedly power-cycled during post fabrication testing, forcing a large number of writes to the on-chip, malicious NV memory. This is done till the malicious NV memory exceeds its write-durability and burns out. Then, in the field, the chip is frequently power-cycled to periodically refresh any malicious, volatile state. Micro-architectural techniques are proposed to allow for power-cycling as frequently as once every 1 million clock cycles without significant performance overhead to the software that runs above.

To date, the design and effectiveness of *non-deterministic* time-based triggers (or **ND-timers**), i.e., timers for which the trigger time is a random variable, have not been addressed. We note that the randomness to which we refer is "true" randomness and is not dependent on the inputs or logic values on the chip, such as the pseudo-random trigger proposed by Wang et al. [26] (see Section 6 for a discussion of the advantages of the proposed ND-timers versus pseudo-random triggers). Randomness has typically been exploited on-chip to enhance security [6]. Ours is, to the best of our knowledge, the first work to demonstrate that true randomness can be harnessed for malicious purposes.

Our Idea The idea behind our proposed ND-timer is simple and powerful. In its simplest form, the ND-timer conducts periodic Bernoulli trials (coin tosses) with a success probability (probability of heads) p, and triggers the attack as soon as a trial is successful (a heads is observed). This results in a *completely state-independent implementation that is immune to power cycling*, but with a wide distribution of trigger time. Next, we observe that the distribution of trigger time can be tightly concentrated around its mean if the attack triggers after k successes instead of the first success. The count must be stored in NV memory but, importantly, we observe that even reasonably small values of k result in very precise trigger times. Hence, both the number of writes and the size of the malicious NV memory can be kept small. At the same time, writes to NV memory are unaffected by power cycling, and therefore our ND-timer cannot be deactivated using post-fabrication power cycling. Finally, compared to D-timers and pseudo-random timers [26], the ND-timer *never* triggers during conventional pre-fabrication simulation based validation.

Contributions Our work is the first to design and evaluate the effectiveness of ND time-based triggers for hardware Trojan activation. The proposed ND-timer is implemented in hardware and has several appealing properties from the perspective of an attacker. We show that ND-timers can be *more* pernicious than traditional deterministic timers in terms of their immunity to state-of-the-art defense mechanisms and design complexity. These advantages, from the attacker's perspective, are highlighted below.

Greater Immunity to State-of-the-Art Defense Mechanisms: We show that ND-timers are immune to a number of state-of-the-art defense mechanisms, for example, the power cycling defense [25], or unused circuit identification [20]. In addition, because our ND-timer harnesses a true, physical source of randomness, it behaves differently in simulation (conventional logic and circuit simulators for digital ICs do not simulate randomness) than it does in the field. ND-timers *never* trigger during logic or circuit simulation, regardless of the length of the simulation, and are therefore challenging to detect during pre-fabrication validation tests. Deterministic timers, including the pseudo-random triggers proposed by Wang et al. [26], do not have this property.

Low Design Complexity: We show that an ND-timer can be implemented using comparable hardware resources as their deterministic counterparts. Furthermore, compared to a deterministic timer, the proposed ND-timer makes use of far fewer bits of NV memory while being equally, or more, effective. For instance, a candidate implementation of the ND-timer uses 14-bits of NV memory only, while a D-timer with similar immunity to power cycling requires at least 3000 bits. The ND-timer that we propose uses standard circuit components that are ubiquitously found in modern digital circuits, making it easy to hide an ND-timer based trigger in regular circuitry.

Precise Control over Trigger Time: We show that ND-timers can be designed such that the attacker has very precise control over when the timer triggers in the field, while guaranteeing a negligible probability of the timer triggering during post-fabrication testing. For instance, we show that by appropriate choice of parameters, the probability of the ND-timer triggering can be reduced to values as small as 10^{-100} even if each fabricated chip can be tested for 10 days before shipping to the customer. At the same time, more than 99.5% of the ND-timers trigger within 10 days of the expected trigger time for an attack designed to trigger in one year (see Section 3.3).

In summary, our research contribution is a design, implementation and thorough analysis of ND-timers that can be used to trigger hardware Trojans. Our work is intended to alert hardware verification and test engineers to the considerable threat to the safety of digital ICs from ND-timer based attacks, which have previously received little or no attention because of their perceived impracticality. On the contrary, we show that ND-timers are not only practical, but potentially more pernicious than deterministic timer attacks. A field programmable gate array (FPGA) implementation of the proposed ND-timer which we use to activate hardware Trojans from the TrustHub benchmark suite [1] further highlights its practicality.

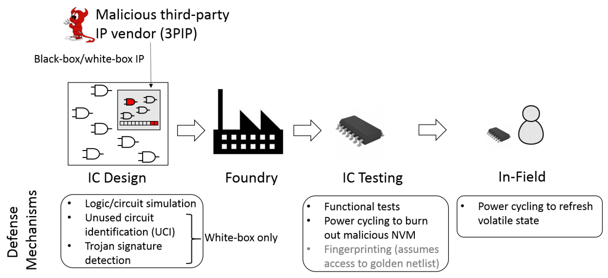
Malicious third-party IP vendor (3PIP)
Black-box/white-box IP
IC Design
Foundry
IC Testing
In-Field
Defense Mechanisms
Logic/circuit simulation
Unused circuit identification (UCI)
Trojan signature detection
White-box only
Functional tests
Power cycling to burn out malicious NVM
Fingerprinting (assumes access to golden netlist)
Power cycling to refresh volatile state

Figure 1: Overview of attack scenario and potential defense mechanisms.

2 Background and Preliminaries

In this section, we describe the threat model that we address in this paper, and the current state-of-the-art in the design of time-based triggers.

2.1 Threat Model

In this section, we first discuss a specific attack scenario in which the proposed ND-timer can be utilized for malicious intent. Next, we describe the capabilities of the defender in mitigating such attacks.

Attack Scenario

Our attack agent is a malicious third-party intellectual property (3PIP) vendor that sells an IP block consisting of digital logic and integrated NV memory. Along with the advertised functionality, the malicious 3PIP vendor inserts a hardware Trojan into its IP block triggerred by a timer. Several 3PIP vendors now sell IP blocks with integrated NV memory. Synopsys, for instance, markets an ultra-low power 90 nm NV memory IP block for use in radio-frequency identification (RFID) and near-field communication (NFC) chips [3]. Malicious 3PIPs that contain hardware Trojans have been acknowledged as a serious security vulnerability for ICs [15, 18].

The 3PIP can be provided to the designer in one of two forms: (a) *Black-box access*, i.e., the designer has no access to the implementation of the 3PIP (for instance, its gate-level or circuit-level netlist) and can only simulate its input/output functionality. Or, (b) *white-box access*, i.e., the designer has full access to the 3PIP down to its circuit-level netlist. The designer can not only apply inputs and observe outputs, but could also monitor internal wires in the netlist.

In our work, we examine the susceptibility of the ND-timer to state-of-the-art hardware Trojan detection/defense mechanisms under both contractual models. Prior work has asserted that even black-box access to the 3PIP is sufficient to disable deterministic timers using the power cycling defense [25]. This is *not* the case for the proposed ND-timer. In fact, we show that the ND-timer can be effective from the standpoint of a malicious 3PIP vendor even under the white-box model.

2.2 State-of-the-Art: Deterministic Timers

We now discuss the current state-of-the-art in time-based triggers, deterministic timers. We discuss also defense mechanisms to thwart hardware Trojans that are triggered by deterministic timers.

Simple Deterministic Timer that uses Volatile Memory only A deterministic timer triggers after exactly t clock cycles of chip operation. The simplest deterministic timer stores its state, i.e., a count, in on-chip volatile memory. A simple defense against such a timer is to periodically turn off power to the chip. When power is restored, the state in the volatile memory is reset to a default value. This is referred to as power cycling. If the number of clock cycles between successive power cycles, t_p, is less than the trigger time, t, then the timer never triggers.

Enhanced Deterministic Timer that uses NV Memory Prior work [25] considers a more powerful deterministic timer-based trigger that makes use of on-chip NV memory. A drawback of using NV memory is that it has limited write-durability. This means that each bit in NV memory can be written to a limited number of times only, after which it loses its ability to store data. Therefore, an attack, to be practical, must be frugal in the number of updates it makes to its non-volatile state.

To address the limited write-durability of the NV memory, the timer proposed by Waksman and Sethumadhavan [25] uses the following protocol. The state of the counter is stored in volatile memory. However, if the chip is powered off, the volatile state is transferred to NV memory immediately before. It is transferred back to volatile memory when the chip is powered back on.

Proposed Defense Mechanism To defend against this attack, Waksman and Sethumadhavan [25] propose an elegant defense mechanism. The chip is repeatedly power cycled during post-fabrication testing. As the NV memory is written to once in every power cycle, it burns out if the number of power cycles exceeds its write durability. Consequently, in the field, the timer has access to volatile memory only, for which in-field power cycling can be used to periodically refresh the state of the counter, thus rendering it ineffective. The 3PIP is *formally* verified to operate correctly for the period of the power cycle.

The proposed defense mechanism seemingly implies that deterministic timer-based triggers can be silenced using post-fabrication techniques only — a combination of defense at test-time and in-field. We show in this paper that this is not the case. Our proposed non-deterministic timer is immune to both post-fabrication defenses.

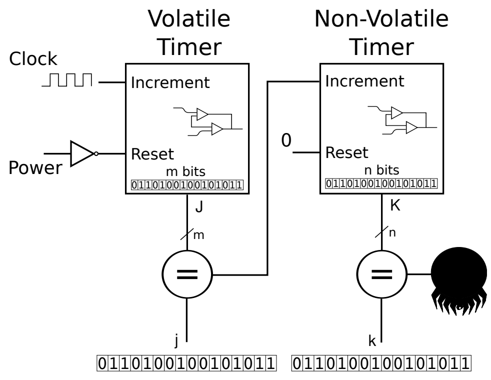

Figure 2: Block diagram of a deterministic timer attack that utilizes a volatile timer chained with a non-volatile timer.

3 Our Attack

In this section, we describe our proposed ND-timer based attack. We begin by proposing an improved deterministic timer design in which the NV memory cannot be worn-out using post-fabrication power cycling as described in Section 2.2. Next, we describe two natural metrics to measure the effectiveness of timer-based triggers that apply to both deterministic and non-deterministic timers: (a) size of the malicious NV memory and (b) the volatile state window. Finally, we describe the ND-timer attack and compare the ND timer with the improved deterministic timer using the two metrics.

3.1 Improved Deterministic Timer Protocol

We now discuss a modification of the deterministic timer-based trigger proposed by Waksman and Sethumadhavan [25], which makes the attack immune to repeated power cycling during post-fabrication testing. In the modified protocol, the NV memory is not written to every time the chip is powered off. Instead, the NV memory is only updated at periodic intervals of time.

A simple implementation of the modified timer comprises both a volatile timer and an NV timer that uses m bits of NV memory. The volatile timer triggers after t_v clock cycles, increments the NV timer and resets. We refer to t_v, the amount of time the count is held in volatile state as the volatile state window (VSW). The attack is triggered when the NV timer reaches a value k, where $k \leq 2^m - 1$. This protocol is illustrated in Figure 2 for clarity. Compared to the deterministic timer of Waksman and Sethumadhavan [25], the NV memory is updated at most k times, regardless of how many times the chip is power cycled. Therefore, the NV memory cannot be burned out during post-fabrication testing. (Of course, we need to ensure that the number of writes to each of the m bits is upper-bounded by the write durability. A sufficient condition is that k is bounded by the write durability.)

While power cycling cannot be used to burn out NV memory in the post-fabrication testing phase, it is still possible to defend against the attack using power cycling in the field. This is another countermeasure proposed by Waksman and Sethumadhavan [25]. In particular, if the number of clock cycles between successive power cycles, $t_p \leq t_v$, the volatile counter will never trigger and the NV memory will never get updated.

Case Study: A Deterministic Timer We now consider a case study that demonstrates that for realistic attack scenarios, even relatively infrequent power cycling can defend against a deterministic timer attack.

We assume that the attacker wants the trigger to occur after the IC operates for one year in the field. We assume also, based on the characterization of 30-nm NAND Flash technology by Cai et al. [7], that the NV memory can tolerate 10,000 program/erase cycles. Based on these assumptions, the NV memory can be written to at most once every 52 minutes. Therefore, power cycling the chip only once every 52 minutes prevents the timer from going off. This case study illustrates that the *limited write durability of NV memory severely limits the effectiveness of deterministic timer based triggers*.

Case Study Continued: Writing More Frequently to NVM What if we want to reduce the time that the count is stored in volatile memory so as to evade the power-cycling counter-measure? To do so, we need more bits of NV memory — in fact, to count up to k with a write-durability of w, we need at least $m \geq \frac{k}{w}$ bits of NV memory.

Let's say we want to force the defender to power-cycle at least one every second, which implies the NV memory will be written to $\approx 3.154 \times 10^7$ in a year. Thus, *at least* 3154 *bits of NV memory are needed to reduce the volatile state window to one second*. This is a $> 200\times$ increase in the number of bits compared to the prior case study where a compact binary encoding was assumed.

Deterministic Timer Based Triggers are Ineffective Even with the improved deterministic timer protocol that we propose, the two case studies above show that the attacker is faced with an unappealing choice. If the attacker wants to keep the number of malicious NV bits small, the deterministic timer has a large VSW and is easily defeated with infrequent power cycling. On the other hand, to force the defender to power cycle the IP frequently, the attacker needs a large number of malicious NV bits.

Using the proposed ND timer, we are able to break this trade-off and implement triggers that require very few NV bits *and* are immune to frequent power cycling.

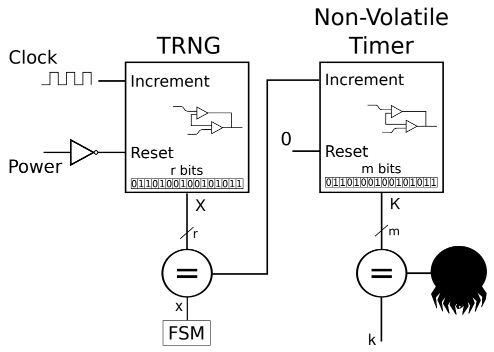

Figure 3: Block diagram of a non-deterministic timer attack that utilizes a TRNG chained with a non-volatile timer. The Finite State Machine (FSM) is used to generate what we call a key to compare with the output of the TRNG.

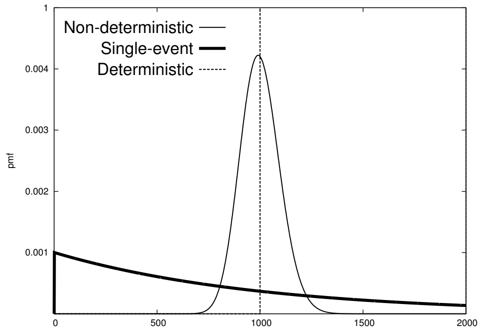

Figure 4: The pmfs (left) for three different timer designs. Each trigger is designed to have an expectation $E(N) = 1000$.

3.2 Proposed Non-Deterministic Timer

In this section, we present the design of our non-deterministic timer. Adopting the mindset of the attacker, we have two design goals. One is to break the apparent trade-off that we discuss in the previous section: between the size of non-volatile storage (NV-memory), and the length of time that volatile state has to be maintained. Our other design goal is to gain control over when the timer triggers, i.e., (a) the timer should not trigger (or trigger with exceedingly low probability) during post-fabrication testing, and (b) minimize the standard deviation (spread) of trigger time around the expected trigger time.

Our timer works as follows. We maintain a counter K in NV memory to keep track of when the timer should trigger, i.e., when K reaches a pre-defined value k, the timer triggers. However, we do not use a periodic clock to increment a counter, as in deterministic timers. Instead, we assume access to a source of randomness which allows us to conduct a series of Bernoulli trials. Each such trial has one of only two outcomes: success with probability p or failure with probability $1-p$. We increment K if and only if we have a successful trial.

Each of our Bernoulli trials is independent of the others, and their probabilities are identically distributed. The probability of success, p, is configurable and the choice of p is a key design parameter for the ND timer, as will be discussed shortly. In Section 3.3.2 we will discuss how we realize the Bernoulli trials in practice, but focus on analyzing the properties of the ND-timer attack for now.

We begin by noting that other than the counter K, we need to maintain no state. The only actions we perform between increments of K are the Bernoulli trials, and a check of the outcome of each to see if it is a success. Thus the VSW of the ND-timer is simply the time it takes to perform a Bernoulli trial and a check of the outcome. As we will see, this provides the ND-timer with greater immunity to in-field power cycling than comparable D-timer implementations.

Single-event trigger To explain our design in more detail, we first consider the somewhat simpler objective of a single-event trigger. That is, a timer that triggers after one successful Bernoulli trial.

Let $N \in [0,\infty)$ be the random variable that measures the number of trials up to and including the first success. We consider what the Probability Mass Function (pmf) $f(n)$, expectation $E(N)$, variance $V(N)$ and standard deviation σ are for the random variable N. It can be shown that: $f(n) = p(1-p)^{n-1}, E(N) = \frac{1}{p}, V(N) = \frac{1-p}{p^2}$, and, $\sigma \approx \frac{1}{p}$, for small p. For small p, the ratio $\sigma/E(N) \approx 1$ which indicates a wide distribution around the expected value.

As an example let $E(N) = 1000$, $p = 0.001$. In Figure 4, we show the corresponding pmf of trigger time. The single event trigger has major drawback from the perspective of the attacker: it has a high probability of detection during the post-fabrication testing phase. For example, if the testing phase lasts for the first 100 trials, there is a 9.5% chance of a timer being triggered in this phase, thus compromising the attack. Even worse for the attacker, if each fabricated IC is tested, the likelihood that the attack is detected in at least one IC is even higher. If even 100 ICs are tested, the attack is detected with probability 0.999.

We now discuss the multiple event trigger which results in a more favorable distribution of attack time in terms of emulating a deterministic trigger and reducing the likelihood of early detection.

Multiple-event trigger We now analyze the statistics of waiting for k successes, instead of a single success, before triggering the attack. Let N_i be a random variable that represents number of Bernoulli trials between the $(i-1)^{th}$ and i^{th} success. As before, let N represent the total number of trials before k successes. We know that:

$$N = \sum_{i=1}^{k} N_i$$

N_i, for all $i \in [1,K]$, are independent and identically distributed random variables with a distribution that was derived in the preceding discussion on the single event trigger. As a consequence of the Central Limit Theorem [17], the distribution of random variable N will, as $k \to \infty$, tend to a Normal distribution with expectation

$$E(N) = kE(N_i) = \frac{k}{p}$$

and standard deviation

$$\sigma_N = \sqrt{k}\sigma_{N_i} = \frac{\sqrt{k}}{p}$$

We observe that standard deviation as a percentage of the expected number of trials after which the attack triggers depends on the value of k only. In particular, $\frac{\sigma_N}{E(N)} = \frac{1}{\sqrt{k}}$. Thus, large values of k result in a narrow distribution around the expected trigger time, while smaller values of k result in a wider distribution. Thus, k is a knob that the attacker can control to effect an attack that either resembles a deterministic attack (large k), or appears to be seemingly random.

Figure 4 shows the pmf of attack time for a multiple event non-deterministic timer ($k = 100$, $p = 0.1$ and $E(N) = 1000$), along with the deterministic and single event trigger distributions for reference. Observe from the figure, that the pmf of the multiple event timer is narrowly concentrated around the desired mean value. Compared to a single event trigger, the multiple event trigger has a much lower likelihood of early detection. As before, if 100 trials are used in post-fabrication testing, the probability that the attack is detected on any given IC is only 10^{-100}. Thus, even if 100,000 ICs are fabricated and each is tested, the attack is detected with probability of only $\approx 10^{-94}$.

We note that, in the discussion so far, we have used the abstract notion of trials as a measure to time. This is easily converted to physical units by noting the time period (in seconds) between successive trials. We discuss this in more detail in the next section.

3.3 FPGA Realization of ND-Timer

We have implemented a hardware prototype of our non-deterministic timer on an Altera Straix IV FPGA. In our implementation, we use conventional digital logic components only — logic gates and registers that are readily available in any digital IC fabrication process. The Stratix IV FPGA does not have on-chip NV memory; we use conventional volatile memory blocks instead. In practice, we expect that an NV memory technology, such as NAND Flash, which is compatible with a standard IC fabrication process, will be used to implement the timer.

Two components of the implementation merit further discussion. Firstly, although the logic required for the attack is straightforward, i.e., comparison of a random number to a static key (see Figure 3), we also propose an advanced implementation of the attack that uses a *dynamic key* to evade recently proposed defense mechanisms such as unused circuit identification [10]. Second, we describe briefly our TRNG implementation, based on the work of Sunar et al. [21], and the subsequent modification due to Wold and Tan [27], that exploits oscillator phase noise.

3.3.1 Dynamic Keys

Since our attack exploits physical sources of randomness that are not modeled by circuit simulators, our attack remains dormant in pre-fabrication simulations — in fact, it never triggers in simulation. While this can be viewed as an advantage, there are defense mechanisms such as UCI [10] that identify unused circuits by flagging pairs of wires that hold the same logic value in simulation. Such a technique will identify the key and count registers as dormant in simulation and flag them.

Although UCI has been defeated [20] (and the technique in that work can be used by us as well to defeat UCI) for the ND-timer attack, provides new opportunities to evade logic-signature based defense mechanisms since, as observed above, the circuit behaves differently in simulation than in the field.

In particular, instead of implementing a simple static key, we instead implement a dynamically changing key that is generated by a (simple) Finite State Machine (FSM), two examples of which are illustrated in in Figure 5. In the first example, for instance, the count register is updated once (only in simulation) while the key value is updated along with count. In fact, any FSM that results in at least one and at most $K-1$ key matches with the TRNG output *in simulation* is a candidate implementation and results in between one to $K-1$ updates to the NV state, count. Since neither the key nor count variables are dormant in simulation, the dynamic key implementation additionally evade detection by UCI and similar techniques.

3.3.2 TRNG Implementation

Figure 6 shows a circuit diagram of our TRNG implementation based on prior work [21, 27]. As seen, the TRNG utilizes 16 parallel ROs, each consisting of a ring of three inverters, that generate noisy clock signals. The RO clocks are sampled by a 50 MHz system clock using a flip-flop. This results in a stream of bits synchronized to the system clock. The outputs of the sampling flip-flops are combined together using an exclusive-or (xor) gate. Finally, the bit-stream at the output of the xor gate is decimated, i.e., each chunk of 1024 bits is xor-ed to produce only one random bit at the output. As shown in

```
case(state)
      ⋮
  SAMPLE :
  begin
    key = count[7:0];
    if (rand[7:0] == key)
    begin
      count <= count + 1;
    end
    state <= DECIMATE;
  end
```

```
case(state)
  ⋮
  SAMPLE :
  begin
    if (count < 5 || key == 0)
        key <= key + 1;
    if (rand[7:0] == key)
    begin
      count <= count + 1;
    end
    state <= DECIMATE;
  end
```

Figure 5: Two possible choices for dynamic key implementation: the current value of the count, and some function of the count and the current value of the key.

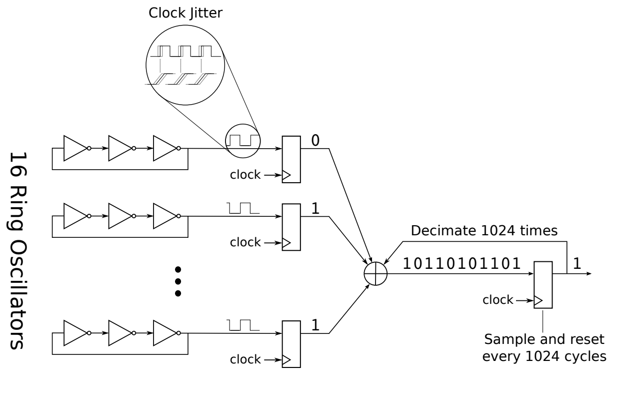

Figure 6: Circuit diagram of the TRNG that we implement.

Figure 6, only a single xor gate is used to perform 1024:1 decimation. In Section 4, we use the National Institute of Standards and Technology (NIST) statistical test suite for random number generators to validate the quality of the random bit-stream that we obtain from our TRNG implementation and show that it passes all NIST mandated tests. At least 16 parallel ROs and 1024:1 decimation is required to pass all NIST tests. Designs with fewer parallel ROs or lesser decimations failed one or more tests.

Given a random bit stream, the match succeeds with probability $p = \frac{1}{2^r}$. As each random bit take 1024 clock cycles to generate and r bits are matched in each Bernoulli trial, the volatile state window, t_v, for the non-deterministic timer is $1024r$ clock cycles. In other words, power cycling the chip at a rate faster than once every $1024r$ clock cycles ensures that the non-deterministic timer never triggers. We show, however, that in practice, the volatile state window is small. For practical attack scenarios, defending against the attack requires the chip to be power cycled once every 16,000–32,000 clock cycles. To put this in perspective, we note that for an IC running at 1 GHz, 16,000 clock cycles corresponds to only 16 μs of time. In other words, the IC will have to be power cycles every 16 μs to defend against such an attack.

3.4 Case Study

The choice of r and k is governed by the following factors:

- The expected trigger time, t, which is, $t = \frac{k}{p} = k2^r$.
- The standard deviation of trigger time, σ_t, which is, $\sigma_t = \frac{\sqrt{k}}{p} = \sqrt{k}2^r$.
- Assuming a binary encoding of the count stored in NV memory, k should be less than the NV memory write durability, w. Therefore, $k \leq w$.
- The volatile state window, t_v. For our implementation, $t_v = 1024r$.

As a basis for comparison, we study the realization of a non-deterministic timer attack which triggers (in expectation) after one year. As before, we assume an NV memory write durability of 10,000 program/erase cycles. The same specifications were used for the deterministic timer study in Section 3.1.

Assuming a 1 GHz system clock, the attack can be implemented using $r = 27$ and $k = 8498$. The attack has an expected trigger time of one year, a standard deviation of ≈ 3.96 days, a volatile state window of only $27.6\mu s$, and requires 14 bits of NV memory. Even if every fabricated chip is tested for 10 days, the probability of the timer triggering during post-fabrication testing is only 10^{-100}. If $100,000$ chips are tested, the probability of at least one timer trigerring is still only 10^{-94}.

As we discuss in Section 3.1, the same attack implemented with a deterministic timer has a volatile state window of 52 minutes for the same amount of NV writes, or requires at least a 136 MB NV memory for the same volatile window.

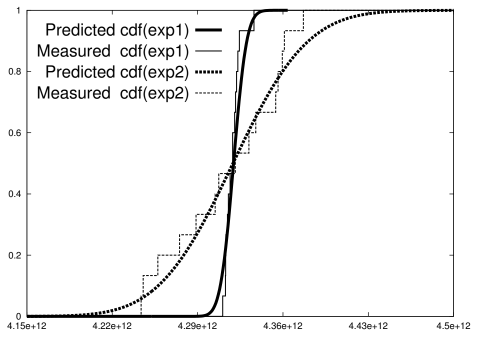

Figure 7: Measured and predicted cdfs for the timer experiments.

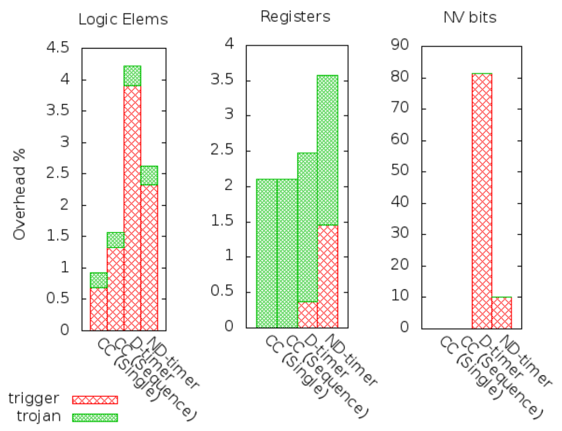

Figure 8: FPGA resource utilization overhead for four different trigger designs for an AES implementation with a CDMA Trojan from the TrustHub benchmark suite.

4 Experimental Results

We now discuss our experimental results obtained from the FPGA prototype of the non-deterministic timer. Our TRNG implementation passes all the standard NIST tests [19]. At the 50 MHz FPGA clock, throughput of 48.8 Kbits/second. For a practical implementation of the attack in a digital IC with a 1 GHz clock, for example, the throughput will be $\approx$ 1 Mbits/second. We present data from a series of attacks in which we programmed the non-deterministic timer prototype to trigger after 24-hours. We present also the hardware resource utilization and power consumption of the non-deterministic timer used as a trigger for Trojans from the TrustHub benchmark suite [1], compared to conventional cheat code (CC) and deterministic timer based triggers.

4.1 24-Hour Attacks

We expect real timer-based attacks on ICs to have an expected trigger time in the order of months or even years. However, to obtain statistically significant data, we need to repeat the same experiment many times. Therefore, it was impractical for us, given limited FPGA resources, to experiment with month or year-long attacks.

To validate the proposed attack, we ran experiments in which the expected trigger time is set to 24 hours. To illustrate the ability of the attacker to control the pmf and cdf of trigger-time, we ran two experiments with the same expected trigger-time but different standard deviations. In Experiment 1, the standard deviation is set to 2.84 minutes to emulate a more deterministic attack, while in Experiment 2, the standard deviation is set to 16 minutes. Each experiment is repeated 15 times to provide enough data for a statistical hypothesis test. The results of these experiments are shown in Figure 7, and we have used hypothesis testing to validate that the sample data is consistent with the distributions predicted analytically.

4.2 Area and Power Footprint

To analyze the footprint of the proposed ND-timer, we compared the area and power overhead of the ND-timer to that of other attacks in two different attack scenarios.

AES Trojan from TrustHub In the first experiment, we consider a benchmark from the TrustHub benchmark suite [1] that consists of a hardware Trojan inserted in an AES module. The Trojan uses code division multiple access (CDMA) concepts to leak the secret key over several clock cycles. In this benchmark, the Trojan is triggered using cheat codes (CC): a single CC (AES-T1000) and a sequential CC over multiple clock cycles (AES-T1100).

To evaluate time-based triggers, we replaced the CC trigger with both a D-timer and our proposed ND-timer. Both timers have a trigger time of 1 year, but the D-timer has a significantly larger VSW compared to the ND-timer. That is, the D-timer is more susceptible to the power cycling defense mechanism. We note that a D-timer with the same VSW as the ND-timer would have an impractically large logic and NV memory footprint. Thus, if anything, our results are biased in favor of the D-timer.

Figure 8 shows the overhead in the number of logic elements (LEs), registers and NV bits of the Trojan and trigger circuits compared to the AES module. Note that although the two timer-based triggers have slightly greater overheads compared to CC triggers, the overall footprint of the attack is less than 4.5% of the AES footprint in all cases.

Comparing the ND-timer to the D-timer, we note that the ND-timer requires fewer LEs but more registers. However, we note that each LE constitutes multiple equivalent logic gates [2]. In addition, the proposed ND-timer requires far fewer NV bits than the D-timer, while also being immune to the power cycling defense.

Power (mW)	AES	Trojan	Trigger
CC (single)	67.64	0.15	0.00
CC (sequence)	67.62	0.15	0.00
D-Timer	74.08	0.25	0.16
ND-Timer	67.99	0.19	0.19

Table 1: Power consumption of the AES module, the Trojan and the trigger as reported by the Altera PowerPlay power estimation tool for four different triggers: CC (single), CC (sequence), D-Timer and ND-Timer.

Table 1 shows the estimated power consumption of the AES block, Trojan and trigger for the four different trigger designs. The power consumption of the two CC triggers is negligible. In contrast, although the D-Timer and ND-Timer do consume some power, it is less than 0.2% of the power consumption of the AES module.

5 Potential Countermeasures

We now discuss some potential countermeasures that can be adopted to defend against the proposed attacks. These include the existing state-of-the-art countermeasures as highlighted in Figure 1, in addition to new potential defenses tailored to our attack.

5.1 Pre-Fabrication Defenses

Pre-fabrication Simulation and Validation Digital circuit simulation tools do not typically capture physical phenomena such as phase noise. From the perspective of the proposed ND timer attack, this is advantageous because the output of the TRNG remains steady at logic zero for the entire length of the simulation, never resulting in a key match. Thus, the ND-timer *never triggers in simulation*, regardless of the length of the simulation. In contrast, deterministic timers and even the pseudo-random timers proposed in [26] will always trigger in simulation, if the circuit is simulated for a sufficiently large amount of time.

Since the ND-timer never triggers in simulation, a possible critique is that it would be detected using techniques like UCI that attempt to identify circuits that are dormant in simulation. We note, however, UCI has been defeated by [20], and our attack can also use this approach to avoid detection by UCI. In addition, dynamic keys, as discussed in Section 3.3, can be used to *further* enhance the immunity of our attack to UCI and related defense mechanisms. The simplest such implementation is one in which the key increments in every clock cycle, thus ensuring that the key register is never dormant. In addition, as shown in Figure 5, the count register can also be made to update (in simulation) up to any value that is less than the count that actually sets off the trigger. Dynamic keys also help evade more recent techniques such as FANCI that look for "nearly unused" circuits. A recent approach defeats FANCI by "spreading the trigger logic into multiple sequential levels" so that it is "not easily differentiable from normal logic" in terms of activity [30]. Again, the proposed ND-timer can borrow from these techniques to further evade FANCI.

Detecting ND-Timer Signatures Security validation teams could search for tell-tale signatures of our attack: counters, comparators, ROs and NV memory bits. However, this is not a simple task — these building blocks are ubiquitous in most chips (including ROs that are widely used for on-chip temperature and process monitoring).

Furthermore, the ND-timer attack (and, in fact, also the D-timer) can be implemented in many different ways. This has already been illustrated in Figure 5. In addition, FSM obfuscation techniques [8, 14] can be used to evade signature detection. Several different on-chip TRNG implementations exist.

With regards to detecting malicious NV bits, as mentioned in Section 2, our attack is specific to IP blocks that already contain integrated NV memory. Thus the mere presence of NV memory in the IP block is, by itself, not sufficient to indicate the presence of a hardware Trojan. In fact, even extra NV bits in the IP will not necessarily raise a red flag, since spare NV bits are typically inserted in NV memories for reliability purposes [5]. Since the proposed ND-timer requires very few NV bits (¡14 bits for a 1 year attack), the attacker can leverage spare or unused NV bits to implement the attack.

5.2 Post-Fabrication Defenses

IC Fingerprinting and Run-Time Signature Detection IC fingerprinting [4, 12, 28] compares the measured characteristics of fabricated ICs, power consumption and timing for instance, to "golden" values.

However, as noted in Section 4, the ND timer is tiny (it uses less than 1% of the logic resources used by a simple RISC processor). — modern day digital ICs can consist of tens or even hundreds of such processors. Furthermore, the power consumption of the ND-timer is estimated to be only 0.2% of that of a Trojan-free AES block. The power overhead will be even lower when compared to larger blocks. Also, since triggers only feed into the Trojan and not the actual circuit under attack, they have no impact on the circuit's delay characteristics. Thermal imaging can be also be used to detect malicious circuits with high power density. However, our results indicate that the power density (power/area)

Electro-magentic (EM) scanning is another side-channel that can be used to detect hardware Trojans, but can be confounded by the presence of other non-malicious ROs in the IP block, or by setting the oscillation frequency of the malicious ROs close to the clock frequency of the chip.

Finally, note that the ND-timer is stealthy in its use of NV memory — it updates NV memory only sparingly. For instance, the 1-year trigger described in Section 3.3 only updates NV memory once every hour, on average.

Disabling the TRNG Techniques to disable the TRNG including cooling to very low temperatures [9], or using frequency injection in the power supply to disrupt the TRNG [16]. However, the former would be impractical to deploy in the field, and the both techniques would disrupt the operation of non-malicious TRNGs on the chip. Nonetheless, these are potentially promising defense mechanisms that merit further investigation. Increasing the TRNG temperature, on the other hand, increases entropy and results in a bit-stream with statistics that are even closer to a true random bit-stream. Thus the attack statistics will follow the analytically predicted distribution even more closely.

5.3 Discussion

In light of the preceding analysis, it appears that existing counter-measures do not adequately guard against the threat of hardware Trojans triggered using ND-timers. The problem is that ND-timer triggered Trojans are *stealthy*; they remain dormant during post-fabrication testing and are hard to disable in the field. Recent work has demonstrated, in actual silicon, a stealthy data-dependent hardware Trojan triggering mechanism [29]. How can the threat from such stealthily triggered Trojans be mitigated? One promising solution is for untrusted chips to provide proofs-of-correctness (at run-time) for *each* computation they perform [23]. However, the costs of generating (and verifying) these proofs are relatively high. Whether the cost can be reduced so as to make this approach practical for commercial deployment remains an open question.

6 Related Work

There is a considerable body of prior work on hardware security. Our work pertains to the design of a non-deterministic timer that can be used to trigger a maliciously inserted hardware backdoor.

Three types of triggers are discussed in prior work [25]: data based, timer based and hybrid triggers. Jin et al. [11] describe the outcome of a competition in which teams of researchers were tasked with inserting hardware attacks in the HDL description of a cryptographic device. The hardware Trojans that Jin et al. describe all utilize either a data- or time-trigger to enable the malicious backdoor circuit.

Data-based triggers monitor internal wires of the circuit for specific bit patterns, and trigger when this pattern is detected. King et al. [13] describe an attack that is triggered by the receipt of an unsolicited network packet containing a "magic" byte; this is an example of a data based trigger. Recently, Zhang et al. have proposed stealthy data-based triggers that defeat several state-of-the-art Trojan detection mechanisms. Very recently, Yang et al. presented a data-based trigger using an ingenious analog circuit implementation [29]. However, data-based triggers require collusion to insert the cheat code, for example, by exploiting a software vulnerability. Timer-based triggers, on the other hand, are entirely self-contained.

Timer-based triggers can be either deterministic [25], pseudo-random [26], or non-deterministic (as in this work). Wang et al. [26] quantify the hardware resource utilization of a number of different trigger implementations including a D-timer and a hybrid "pseudo-random" data and time trigger. This trigger uses uncommonly occurring data patterns in the design to increment its counter instead of counting up in every clock cycle. Although the trigger time can vary depending on how frequently these uncommon inputs appear, the "randomness" that is being exploited is that of the users' behaviour (which determines the inputs to the chip). Unlike our ND-timer where the randomness is precisely characterized and results in an accurate characterization of the distribution of trigger time, randomness in user behaviour is poorly characterized and can vary widely from one user to another. For instance, are user inputs independent or correlated with time? Thus, unlike the proposed ND-timer attack, the distribution of trigger time for the "pseudo-random" timer is difficult to characterize. Furthermore, unlike our attack, the pseudo-random timer *triggers in simulation* if the appropriate inputs are applied. The ND-timer never triggers in simulation since conventional simulators do not model physical randomness. The pseudo-random trigger also *triggers during post-fabrication testing*, again assuming the right inputs are applied. If these inputs are used on every tested chip (as is typically the case), the Trojan would be detected because the attack would trigger in each case. On the other hand, The ND-timer triggers with negligibly low likelihood ($< 10^{-94}$) in post-fabrication testing, and even in the very unlikely event that an attack does trigger on a chip under test, it would be discarded along with others that fail because of random defects.

7 Conclusion

In this paper, we have analyzed and designed non-deterministic timer-based triggers for hardware Trojan activation, and demonstrated that they have several appealing properties from the perspective of the attacker. The proposed ND-timers are immune to previously proposed power cycling based defense mechanisms. Although non-deterministic, an attacker can control the time at which the timer triggers with surprising precision. Furthermore, because they harness true, physical sources of randomness, ND-timers behave differently in simulation than they do in the field. We show how attackers can leverage this property to their advantage. Finally, based on an FPGA prototype we show that ND-timers are not only practical, but also have a negligible area and power footprint. In future work, we will investigate mechanisms to defeat ND-timer based attacks.

References

[1] Trust-hub. `https://www.trust-hub.org/resources/benchmarks`.

[2] Altera Stratix IV. `https://www.altera.com/en_US/pdfs/literature/br/br-stratix-iv-hardcopy-iv.pdf`, 2013.

[3] Synopsys New Ultra Low-Power Non-Volatile Memory IP Cuts Power by 90 Percent and Size in Half. `http://news.synopsys.com/`, 2013.

[4] AGRAWAL, D., BAKTIR, S., KARAKOYUNLU, D., ROHATGI, P., AND SUNAR, B. Trojan detection using ic fingerprinting. In *Security and Privacy, 2007. SP'07. IEEE Symposium on* (2007), IEEE, pp. 296–310.

[5] BREWER, J. E., AND GILL, M. *Nonvolatile memory technologies with emphasis on flash*. Wiley, 2008.

[6] BUCCI, M., GERMANI, L., LUZZI, R., TRIFILETTI, A., AND VARANONUOVO, M. A high-speed oscillator-based truly random number source for cryptographic applications on a smart card ic. *Computers, IEEE Transactions on 52*, 4 (2003), 403–409.

[7] CAI, Y., HARATSCH, E., MUTLU, O., AND MAI, K. Error patterns in mlc nand flash memory: Measurement, characterization, and analysis. In *DATE, 2012* (2012), IEEE, pp. 521–526.

[8] CHAKRABORTY, R. S., AND BHUNIA, S. Harpoon: an obfuscation-based soc design methodology for hardware protection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 28*, 10 (2009), 1493–1502.

[9] GILDENBLAT, G., COLONNA-ROMANO, L., LAU, D., AND NELSEN, D. Investigation of cryogenic cmos performance. In *Electron Devices Meeting, 1985 International* (1985), vol. 31, IEEE, pp. 268–271.

[10] HICKS, M., FINNICUM, M., KING, S. T., MARTIN, M. M. K., AND SMITH, J. M. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *IEEE Symposium on Security and Privacy* (2010), pp. 159–172.

[11] JIN, Y., KUPP, N., AND MAKRIS, Y. Experiences in hardware trojan design and implementation. In *Hardware-Oriented Security and Trust, 2009. HOST'09. IEEE International Workshop on* (2009), IEEE, pp. 50–57.

[12] JIN, Y., AND MAKRIS, Y. Hardware trojan detection using path delay fingerprint. In *HOST 2008* (2008), IEEE, pp. 51–57.

[13] KING, S., TUCEK, J., COZZIE, A., GRIER, C., JIANG, W., AND ZHOU, Y. Designing and implementing malicious hardware. In *Proceedings of the 1st USENIX Workshop on Large-scale Exploits and Emergent Threats* (2008), USENIX Association, pp. 1–8.

[14] KOUSHANFAR, F., AND ALKABANI, Y. Provably secure obfuscation of diverse watermarks for sequential circuits. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on* (2010), IEEE, pp. 42–47.

[15] LIU, C., RAJENDRAN, J., YANG, C., AND KARRI, R. Shielding heterogeneous mpsocs from untrustworthy 3pips through security-driven task scheduling. In *Defect and Fault Tolerance, IEEE International Symposium on* (2013), IEEE, pp. 101–106.

[16] MARKETTOS, A. T., AND MOORE, S. W. The frequency injection attack on ring-oscillator-based true random number generators. In *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 317–331.

[17] PAPOULIS, A., AND PROBABILITY, R. *Stochastic processes*, vol. 3. McGraw-hill New York, 1991.

[18] SHRESTHA, G., AND HSIAO, M. Ensuring trust of third-party hardware design with constrained sequential equivalence checking. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for* (2012), IEEE, pp. 7–12.

[19] SOTO, J. Statistical testing of random number generators. In *Proceedings of the 22nd National Information Systems Security Conference* (1999), vol. 10, NIST Gaithersburg, MD, p. 12.

[20] STURTON, C., HICKS, M., WAGNER, D., AND KING, S. Defeating uci: Building stealthy and malicious hardware. In *IEEE Symposium on Security and Privacy* (2011), pp. 64–77.

[21] SUNAR, B., MARTIN, W., AND STINSON, D. A provably secure true random number generator with built-in tolerance to active attacks. *Computers, IEEE Transactions on 56*, 1 (2007), 109–119.

[22] TEHRANIPOOR, M., AND KOUSHANFAR, F. A survey of hardware trojan taxonomy and detection. *Design & Test of Computers, IEEE 27*, 1 (2010), 10–25.

[23] WAHBY, R. S., HOWALD, M., GARG, S., AND WALFISH, M. Verifiable asics.

[24] WAKSMAN, A., AND SETHUMADHAVAN, S. Tamper evident microprocessors. In *Security and Privacy (SP), 2010 IEEE Symposium on* (2010), IEEE, pp. 173–188.

[25] WAKSMAN, A., AND SETHUMADHAVAN, S. Silencing hardware backdoors. In *IEEE Symposium on Security and Privacy* (2011), pp. 49–63.

[26] WANG, X., NARASIMHAN, S., KRISHNA, A., MAL-SARKAR, T., AND BHUNIA, S. Sequential hardware trojan: Side-channel aware design and placement. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on* (2011), IEEE, pp. 297–300.

[27] WOLD, K., AND TAN, C. Analysis and enhancement of random number generator in fpga based on oscillator rings. *International Journal of Reconfigurable Computing 2009* (2009), 4.

[28] WOLFF, F., PAPACHRISTOU, C., BHUNIA, S., AND CHAKRABORTY, R. Towards trojan-free trusted ics: Problem analysis and detection scheme. In *DATE, 2008* (2008), IEEE, pp. 1362–1365.

[29] YANG, K., HICKS, M., DONG, Q., AUSTIN, T., AND SYLVESTER, D. A2: Analog malicious hardware, 2016.

[30] ZHANG, J., YUAN, F., AND XU, Q. Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In *Proceedings of CCS* (2014), ACM, pp. 153–166.