On Borrowed Time – Preventing Static Power Side-Channel Analysis

Robert Dumitru
The University of Adelaide &
Defence Science and Technology Group
robert.dumitru@adelaide.edu.au

Andrew Wabnitz
Defence Science and Technology Group
andrew.wabnitz1@defence.gov.au

Yuval Yarom[§]
Ruhr University Bochum
yuval.yarom@rub.de

***Abstract*—In recent years, static power side-channel analysis attacks have emerged as a serious threat to cryptographic implementations, overcoming state-of-the-art countermeasures against side-channel attacks. The continued down-scaling of semiconductor process technology, which results in an increase of the relative weight of static power in the total power budget of circuits, will only improve the viability of static power side-channel analysis attacks. Yet, despite the threat posed, limited work has been invested into mitigating this class of attack.**

In this work we address this gap. We observe that static power side-channel analysis relies on stopping the target circuit's clock over a prolonged period, during which the circuit holds secret information in its registers. We propose Borrowed Time, a countermeasure that hinders an attacker's ability to leverage such clock control. Borrowed Time detects a stopped clock and triggers a reset that wipes any registers containing sensitive intermediates, whose leakages would otherwise be exploitable.

We demonstrate the effectiveness of our countermeasure by performing practical Correlation Power Analysis attacks under optimal conditions against an AES implementation on an FPGA target with and without our countermeasure in place. In the unprotected case, we can recover the entire secret key using traces from 1,500 encryptions. Under the same conditions, the protected implementation successfully prevents key recovery even with traces from 1,000,000 encryptions.

1. Introduction

The seminal work of Kocher [29] demonstrated that implementations of mathematically secure cryptographic primitives can be vulnerable to attack via side-channel analysis exploiting the leakage of sensitive information through physical properties of the implementation. Since then multiple side channels have been demonstrated, exploiting various effects, such as timing [9, 12], power consumption [28, 35], electromagnetic emanations [17, 51], shared micro-architectural components [18, 58], and even acoustic [19] and photonic emanations [32, 53]. Among the purely physical sources of side-channel leakage, exploitation of power consumption, known as power analysis, has received the most attention from the security community.

§. Work partially done while affiliated with The University of Adelaide.

Research on power analysis has historically focused on attacks exploiting the instantaneous power consumed while performing computations, also known as dynamic power analysis. However, with the continued down-scaling of Complementary Metal-Oxide-Semiconductor (CMOS) technology the relative weight of static power, used for maintaining the logical state of a circuit, has increased as part of the total power budget, and with it some attention has more recently shifted to static power side-channel analysis [4, 8, 27, 38, 39, 41, 42, 43, 50].

In a typical static power side-channel analysis attack, the attacker freezes the state of the target device when it is known to contain secret data in some of its state registers. The attacker then waits for some time to allow the dynamic effects of prior computation to subside, before measuring the power consumption of the device. The attacker typically performs a large number of measurements (e.g. 100k samples [41]) over a measurement period. Averaging these measurements significantly reduces measurement noise, allowing high measurement accuracy. Alternatively, the attacker can use measurement devices that provide high precision [4, 40], but we note that those devices have a very low sampling rate (1kS/s), hence measurements still take a relatively long time.

Since the first practical demonstration of static power side-channel analysis attacks in 2014 [43], research in the area has demonstrated attacks on various cryptographic primitives [4, 41], investigation of the factors that affect the attack [42], and improvements to attack techniques [4, 38, 41, 42]. In particular, past work has demonstrated that static power side-channel analysis is an effective attack even against targets that implement countermeasures which hinder dynamic power analysis attacks [2, 3, 38, 41]. However, little research has been invested toward designing countermeasures against static power side-channel analysis despite the emergence of a clear need for dedicated solutions. In this work we set out to fill this gap and design an effective countermeasure for this class of attack.

Our Contribution

We first observe that all published static power side-channel analysis attacks require a relatively long period,

spanning at least several hundreds of typical clock cycle periods, in which two conditions hold:

1) The clock signal supplied to the target device is stopped; and
2) Registers contain and thereby leak sensitive data.

The core idea behind our countermeasure, called Borrowed Time, is to prevent situations in which both conditions hold. Specifically, Borrowed Time continuously monitors the clock to ensure that it keeps ticking. If Borrowed Time detects that the clock is stopped for too long, it resets the contents of registers that may contain sensitive data, setting their contents to a fixed, non-secret value.

We propose two approaches for implementing the clock-monitor part of Borrowed Time. One uses clock management circuitry found in conventional digital systems which imposes minimal design overhead, and the other uses a custom asynchronous circuit module that lends itself well to lightweight designs that employ clock-gating. Both designs can be implemented on either FPGA or ASIC devices.

The first approach uses a Phase-Locked Loop (PLL), a component commonly used for clock management in digital systems. In a nutshell, a PLL uses a feedback loop to ensure that an output clock (or clocks), which it distributes across the circuit, remains synchronised with an input clock, typically provided by an external source such as a crystal oscillator or another circuit. Stopping the input clock breaks the synchronisation between it and the output clock, allowing the PLL to rapidly detect such an incident.

While PLLs are effective at detecting a stopped clock, they do have two main limitations. First, PLLs are relatively large and may, therefore, be expensive for some uses. Second, because PLLs take a long time to synchronise the output and input clocks, they are unsuitable in some cases. One such example is within clock-gated circuits, where a master circuit dynamically disables the clock signal provided to some circuit components in order to reduce power consumption when these components are not active. To accommodate for cases where PLLs may not be suitable, we propose an alternative design based on an asynchronous system that samples the clock signal at multiple points in time and monitors its natural variation. Where this variation is absent and the clock value is the same across all sampled points in time, the design indicates a stopped clock.

To test the effectiveness of Borrowed Time, we perform practical static power side-channel analysis attacks with and without the countermeasure in place. We first implement an unprotected AES circuit on a Field-Programmable Gate Array (FPGA) and perform an end-to-end static power side-channel analysis attack on it to recover the secret key. Unprotected implementations exhibit strong leakage that benefits the attacker. With our equipment and attack setup, we can recover the key using 1,500 samples, each being a measured power trace from one encryption, which provides us with a benchmark of the system's information leakage.

Detecting whether a clock has stopped invariably takes some time. At the minimum, the detection mechanism needs to wait and see a missed clock edge. For Borrowed Time to be an effective countermeasure, detection and reset time must be shorter than the time needed to carry out a static power side-channel analysis attack. As mentioned above, the main impediment for carrying out such attacks immediately after stopping the clock is the prolonged period in which dynamic effects from prior state changes affect the power consumption of the target circuit, known as the memory effect. While past works report the wait time (from when the clock is stopped until measurements start to be taken) used for attacks [41, 42], little information is provided in support of the choice of this delay.

Because Borrowed Time relies on the delayed presence of information leakage due to lingering noise from the memory effect, we measure the impact of reducing the wait time on the success of static power side-channel analysis attacks. Reduction of wait time makes attacks harder, and when reduced below 200 µs the attack becomes infeasible. In comparison, Borrowed Time can eliminate leakage by detecting that the clock has stopped and triggering a reset within less than 1 µs, assuming the target hardware is operating in at least the MHz range.

To test the effectiveness of Borrowed Time, we incorporate it in the target design. We demonstrate that both of its implementation variants are effective in preventing the attack, and that with Borrowed Time the target does not show evidence of leakage even with 1,000,000 samples.

To summarise, in this paper we make the following contributions:

- We investigate the memory effect in the context of static power side-channel analysis attacks, demonstrating that the attacker needs to wait a few hundreds of clock cycles after stopping the clock to observe usable static leakage.
- We present two designs of Borrowed Time, a countermeasure for static power side-channel analysis attacks that detects a stopped clock and resets sensitive data within as little as one clock cycle, well before the time window in which an attacker has the opportunity to mount an attack.
- We practically implement both versions of Borrowed Time within an AES hardware implementation and evaluate it by attempting to mount end-to-end key recovery attacks, demonstrating that Borrowed Time effectively mitigates this class of attack.

The rest of this paper is organised as follows. In Section 2 we present necessary background on static power side-channel analysis attacks and digital systems. Section 3 presents the two designs of Borrowed Time. Section 4 describes the setup we use for evaluating Borrowed Time, including the equipment, the target, and the procedure we use. We evaluate the memory effect in Section 5 and Borrowed Time in Section 6. Finally, we discuss the limitations of Borrowed Time and potential future work in Section 7.

2. Background

In this section we provide essential background on static power side-channel analysis attacks and countermeasures, and on digital circuit design concepts that are relevant to our work.

2.1. Physical Side-Channel Analysis

The physical interactions of electronic devices with their environment can leak information about the computations they perform and values processed within. This information leakage manifests through inherent correlations between the computation or data being processed with the observed physical effects. For example, the power dissipated to change a register's contents typically correlates with the number of bit flips between the original and new values being stored. The practice of exploiting such unintended sources of information leakage to reveal secrets is known as physical side-channel analysis. Adversaries seeking to carry out this kind of attack typically require physical access to the target device.

Since the seminal work of Kocher [29], side-channel analysis has become a significant threat to cryptographic implementations in particular, since visibility of computational intermediates defies the black-box assumption[1] that upholds the security of cryptographic primitives. Side-channel attacks are categorised by the effect responsible for leaking information, with significant effort being invested in power analysis, which exploits the power consumption of a target device [28, 35].

Countermeasures. Countermeasure approaches against physical side-channel analysis fall into two main categories: information masking, which increases algorithmic noise; and information hiding, which increases measurement noise.

Masking or 'secret sharing' is a widely adopted class of countermeasures, in which the sensitive values being used for computation within a system are never actually stored in the system at any given moment [13]. For example, a sensitive intermediate bit x is masked by a set of randomly generated mask shares x_0, x_1, ..., x_n, such that $x = x_0 \oplus x_1 \oplus ... \oplus x_n$. The data is 'split' and processed in the form of these shares and combined at the end to produce the output.

Dual-rail pre-charged logic styles [15, 46, 49, 55, 56] are an information hiding class of countermeasures that aim to balance the data-dependence of power dissipation of combinatorial logic gates. They use complementary signals such that exactly one bit is always toggled for each clock cycle transition. Both signal lines are held in the same initial state, for example low, then which one is toggled to high depends on the data. These logic styles are highly dependent on perfectly balanced power consumption from the underlying CMOS transistors and signal routing, which is very difficult to achieve in practice [40].

2.2. Static Power Side-Channel Analysis

The majority of power analysis attacks historically explored are concerned with the dynamic power consumption of target devices, i.e. the instantaneous power consumption associated with changing the logical state of electronic gates. In contrast, static power side-channel analysis, the focus of our work, is concerned with the power dissipated to maintain the logical state of the device, which exhibits a dependence on the data held within [1, 20].

Any digital circuit can be abstracted as a system made up of pipelines of combinatorial logic elements sectioned between sequential (state) elements. The core idea of exploiting static power information leakage is that a target device can be held in a certain state (represented by the contents of all state register elements) by stopping the clock signal fed to the device for a long enough time such that: all dynamic effects from transitioning into the given state have died off, and measurements can be taken across a sufficiently long window to average away a large amount of noise. Static power side-channel analysis considers this averaged univariate random variable (referred to as the measurement *trace*) as representative of a target's given state, whereas dynamic power analysis considers an instantaneous power trace as a time-dependent multivariate random variable representative of a computation performed by the target (which is related to its transition between states). A stronger attack model is typically required by static power side-channel analysis as an adversary must be able to make use of clock manipulation, although Moos [38] showed that in some scenarios attacks based on static power can be performed without outright clock control, instead leveraging the interrupted clock signal provision within a clock-gated target circuit.

The earliest reports on this class of attack were based on simulated analysis [20, 33] and the first practical demonstration was performed by Moradi [43] in 2014 against FPGA targets. Since then, a few more works [4, 8, 27, 38, 39, 41, 42, 50] have practically evaluated static power side-channel analysis attacks. A comprehensive history of the research area is described in [40, 42].

Technology Down-Scaling. Dynamic power analysis has traditionally been the subject of most power side-channel research, however with the continued down-scaling of CMOS technology as a proponent of Moore's law [37], the contribution of static leakage to overall power consumption is becoming proportionally more prominent [26, 45, 63]. Simulations focusing on 90 nm, 65 nm, and 40 nm CMOS technologies show a direct increase in static leakage with smaller scale technologies [33]. From practical results, Moradi [43] demonstrates that the *absolute* leakage current and leakage related to data contents does not necessarily directly increase with smaller technologies between different FPGA families. This work does not however assess the relative proportions of static to dynamic leakage and goes on to mention that there are more differences observed between the different process technology families than simply those related to scale, the details of which are not publicly available. The work also finds logic placement and routing to have a profound impact on static leakage. Moos [38] performs similar testing with ASICs (Application-Specific Integrated Circuits), conclusively demonstrating a drastic increase in data-dependent static leakage across shrinking CMOS technologies.

Measurement Factors. In static power side-channel analysis, while keeping a target circuit in a certain state via

1. Under the black-box model adversaries can only observe inputs and corresponding outputs.

some form of clock control, many samples are taken over a long measurement window which are then averaged into a single value representative of that given target state. This *intra-trace averaging* has been shown to reduce noise exponentially [42].

The most prominent physical effects responsible for quiescent current flow in CMOS circuits are sub-threshold leakage current and reverse biased PN junction current [3]. The amount of leakage current in a system depends on Process, Voltage, and Temperature (PVT) variations. Moos et al. [42] investigate the influence of the magnitude of the voltage and temperature on the measurements, finding that the static current leakage of CMOS hardware exhibits an exponential dependence on operating temperature, while increases in operating voltage also increase the leakage signal but only marginally. Moos [38] shows that the influence of temperature is greater for smaller process technologies. Moreover, the nonlinear temperature dependencies can be leveraged by using the dimension of temperature to conduct multivariate attacks [5, 14] which exploit measurements of the same state over varying temperatures.

Memory Effect and Measurement Interval. To exploit leakage of static register contents an attacker must wait ample time for the dynamic effects within the target circuit to settle before taking measurements. One might therefore expect it to be sufficient to wait until all of the signal value changes from a clock transition are propagated to the end of their paths. However, it turns out that dynamic effects continue to be observable in a circuit's power consumption profile well beyond this time, even extending across several clock cycles in a phenomena coined the *memory effect* in [44]. This effect may arise from slow settling transient response of the leakage current among individual CMOS elements, as well as from aggregated system effects like reflections. Moradi and Mischke [44] use the memory effect to combine leakages from multiple operations across multiple clock cycles into univariate readings. For their FPGA target, the effect that extends their window of observable dynamic leakage vanishes after 4 µs, however additional noise introduced by the memory effect appears to continue beyond that, which would affect static power measurements.

Past works observe that the memory effect is highly influenced by the measurement setup used to acquire power traces [41, 42, 44]. The experimental setups used in these works are very similar and include a high gain ($\times$1000) DC amplifier stage with low (20 kHz) bandwidth. These works mention that from their analysis the memory effect influences static power measurement for the first 20 ms after stopping the clock. The works go into no further detail on how this was evaluated.

Countermeasures. With more attention having been given towards dynamic leakage models than static, similarly the effectiveness of countermeasures has mostly been evaluated by their resistance to dynamic leakage. Thus, widely adopted countermeasures accepted to mitigate power analysis attacks can have shortcomings against a holistic threat model, which includes considering static power side-channel analysis. For example, previous works have indicated that attackers leveraging static power analysis are able to exploit higher-order leakages of masking schemes with much lower data complexity than adversaries seeking to leverage dynamic leakage [38, 41]. In the case of some logic balancing countermeasures, results have even suggested that logic styles resistant to dynamic power analysis can actually heighten vulnerability to static power side-channel analysis [2, 3]. Relatively few countermeasures dedicated to coping with the static power side-channel have been developed.

Time-enclosed logic styles [6, 7, 8, 10] are an extension of dual-rail pre-charged logic that encode data in the time domain such that data is only present for evaluation in a limited *evaluation phase* time window during each clock cycle. Extending on our previous description of dual-rail pre-charged logic, at the end of the evaluation phase time-enclosed logic designs will toggle the signal line that did not previously toggle. This requires an internally derived secondary clock signal to complete the evaluation phase. These countermeasures have the effect of shifting information leakage to higher frequencies as additional switching activity is performed. Bellizia et al. [8] evaluated a time-enclosed logic style as a countermeasure to static power side-channel analysis and found it to be effective in eliminating leakage from combinatorial logic gates, however, information still leaks from state elements.

Several proposed countermeasures [64, 65, 68] take the approach of generating additive noise within the system to reduce the data-dependent signal-to-noise in power leakage.

2.3. Clocking in Digital Systems

Phase-Locked Loops. The most stable and accurate clock signals are generated by crystal oscillators. Clock signals must be distributed to all synchronous elements in a circuit. In order to drive this high fan-out clocking network load, digital designs will typically generate internal oscillatory signals for distribution and synchronise them to an incoming reference signal from a crystal oscillator. A Phase-Locked Loop (PLL) is the closed-loop control system used to achieve this synchronisation. PLLs are found in many digital systems within their clock management primitives, they are commonly used for clock generation, timing distribution, clock recovery, frequency synthesis, and frequency demodulation.

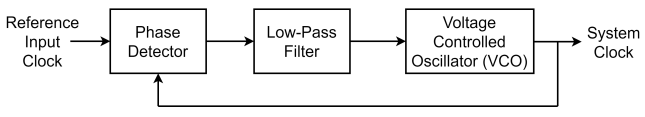

Figure 1. Block diagram showing basic elements of a PLL

The basic components of a PLL are shown in Figure 1. The clock signal distributed throughout the circuit system is generated by a Voltage Controlled Oscillator (VCO), which produces an oscillating signal whose frequency depends on the input voltage. The VCO output is fed back to a Phase

Detector comparator that detects the phase difference between the system and reference clocks. This phase difference acts as an error signal fed forward to the VCO in order to change its frequency to match the reference. A Low-Pass Filter stage removes unwanted high frequency noise from the Phase Detector.

Clock-Gating. Clock-gating is a design methodology in which the clock fed to part of a circuit is only enabled while that part is in use. This significantly reduces the dynamic power dissipation of the gated circuit at the relatively low cost of implementing additional logic to enable and disable the clock. Within smartcards this design practice has been shown to reduce total power consumption by up to 40% [67]. Clock-gating a circuit with a PLL-driven clocking network is not viable however, as PLLs require a stable incoming clock before they can achieve lock.

Clocking in Side-Channel and Fault Injection Attacks. Another popular class of attack that involves clock signal manipulation are clock glitching fault injection attacks. In these attacks rapid glitched clock pulses are injected between regular pulses which violate a circuit's timing constraints. This can be used to bypass certain security-critical operations or instructions by not giving them sufficient time to terminate between pulses. Using a Digital Clock Manager (DCM) primitive, Luo and Fei [34] simulate a proposed method for detecting high frequency clock signal fault injection. It requires maintaining a higher frequency clock in the target which they use to sample the incoming reference signal.

Clock Sensors. Kömmerling and Kuhn [30] discuss incorporating a robust low-frequency (clock) sensor in the context of protecting smartcards from bus observation using a Scanning Electronic Microscope. Lowering the target devices clock frequency is said to make this kind of e-beam testing easier. They suggest that implementations of filter elements used to detect low-frequency input clocks are commonly found in smartcard processors but are inadequate, and they highlight that defences should be designed to be embedded within processors.

Farheen et al. [16] design a clock-freeze detection sensor and propose it as a countermeasure to Laser Logic State Imaging[2] (LLSI) [31], which they claim may also be applicable to static power side-channel analysis attacks. However, this detection mechanism has only been implemented in isolation, and has not been evaluated as part of a larger design. A limitation of their design is that it does not allow a protected circuit to work within a system where it is clock-gated since their detector takes several cycles to de-assert its detection alarm flag.

Delays and Timing Closure. Timing closure relies on extant well-understood models of signal propagation delays through logic gates and routing elements across PVT operating conditions. These models are also leveraged in ring-oscillator and time-to-digital sensor circuits, in some cases these can be used to carry out side-channel attacks against other modules on the same chip [21, 66], even remotely in cloud logic-renting systems. They can also be deployed as a defensive measure to profile normal system behaviour and monitor for anomalies that may indicate malicious activity [36] such as that of Hardware Trojans [23] or supply voltage attacks [69].

2. A static side-channel analysis technique where instead of observing power consumption, the state of a circuit is visually inspected using a laser, also requires inducing static state in target with a stopped clock condition.

3. The Borrowed Time Countermeasure

Static power side-channel analysis relies on a combination of two major factors. The first being that the adversary must be able to leverage some form of clock manipulation to stop the clock during particular cycles. This can manifest either naturally, during periods of non-activity within a clock-gated target, or as a result of direct adversarial control over the clock signal. The other major factor is that sensitive data must remain present in state register elements for the duration of an extended measurement period under a stopped clock condition.

To address this, we propose an in-chip countermeasure called *Borrowed Time*, to be deployed within a target circuit, that eliminates exploitable static leakage of sensitive data under a stopped clock condition. Borrowed Time involves equipping a target circuit design with a module that monitors the incoming clock signal for a stop condition. Upon detection, the module triggers a reset alarm that clears any sensitive information registered, thereby eliminating the exploitable source of static leakage.

Borrowed Time is specific to the static power side-channel analysis attack class, therefore we assume an adversarial threat model wherein an attacker has the capability to carry out this kind of attack. Specifically, this entails physical access to the chip on which the target resides for taking power measurements and the ability to leverage a stopped clock signal provided to the target.

The Borrowed Time countermeasure is two-fold, first it detects when the target circuit's incoming clock signal is stopped, then it triggers an asynchronous reset of all registers containing sensitive data. Registers with asynchronous resets will set their data contents to zero upon assertion of an asynchronous reset signal, without the need for a clock transition.

We propose two possible approaches for detecting a stopped clock. The first solution involves using standard clock management circuitry that contains a PLL, as found in many conventional digital designs. The second solution comprises a custom asynchronous delay-based clock sampling module, this can be used in conventional designs as well as in clock-gated systems where PLLs will not work. We then propose alteration of the target circuit implementation such that internal registers storing sensitive intermediate data are immediately reset upon a triggered alarm indicating stopped-clock detection, with the aim of preventing information leakage of those intermediates during clock inactivity. Our countermeasure is implemented at the logic design stage and requires no alternative logic style and no alteration in the standard cell libraries.

We now describe our two approaches for implementing Borrowed Time in greater detail.

3.1. PLL-Based Detection

For targets which receive a stable clock signal our clock monitoring solution consists of a PLL. This is a relatively simple solution to implement, as PLL components are often incorporated in both ASIC- and FPGA-based conventional digital systems. This solution is similar in principle to [16] in that an internally generated oscillatory signal is used to check the input clock, however we leverage the engineering effort dedicated towards creating robust clock management systems that incorporate PLLs. When a PLL's reference (input) clock and the feedback clock (output provided to the system) are frequency- and phase-matched, the PLL is said to be locked. To ensure synchronisation, sequential logic elements clocked by a PLL can be held in a reset state until lock is achieved. For this reason clock management modules that contain PLLs will usually provide hardware designers with the option of using output status signals such as a `LOCKED` signal to indicate lock synchronisation. The time a PLL needs to attain a locked state is known as lock time, and is a crucial design parameter among PLLs, generally in the order of microseconds [25, 61].

We propose using the logical NOT of the `LOCKED` signal as a clock manipulation detector. PLLs are typically sensitive enough to de-assert the `LOCKED` signal within one clock cycle of a non-transitioning reference signal [60]. Therefore, by using such a signal to trigger an asynchronous reset we can reduce the period in which data-dependent power consumption is observable to only one immediate additional clock cycle.

As long as an asynchronous reset is asserted for a register, the contents of the register cannot be changed from zero and since an asynchronous reset will be asserted as long as the PLL is not locked, the target circuit module will not be able to resume working normally until a reference clock of the correct frequency is supplied. Further, to avoid metastability problems upon clock resumption the design must perform synchronous de-assertion of the asynchronous reset signal.

The major limitation of a PLL-based solution is that in order to operate correctly, PLLs rely on a stable incoming clock signal over a long period of time which would not be available within a clock-gated target. More generally, any solution for a target within a gated system must not be dependent or reliant on a clock signal with long-term stability.

3.2. Asynchronous Delay-Based Detection

Given the limitation of a PLL-based solution we propose an alternative design that can work under conditions where a continually transitioning clock is not provided. This alternative solution involves incorporating a custom asynchronous module in the target design. It is similarly applicable to both ASIC- and FPGA-based systems, however it has increased implementation complexity compared to the PLL-based solution due to additional design considerations. Incorporation of this solution into a target design will allow the module to operate as normal in both gated and non-gated systems, while also protecting against attacks that exploit a gated stopped clock condition [38].

The clock-monitoring module consists of a chain of unit delay elements that takes the incoming clock signal as the input. See Figure 2. The output of each element is a version of the clock, shifted by the propagation delay time through the element. A subset of these time-shifted versions of the clock are fed as input to a combinatorial logic circuit that goes high if all of its inputs are equal. We refer to each input signal line fed to this circuit as a tap on the chain of delay elements. Table 1 shows the truth table for the combinatorial element. If all inputs to the combinatorial gate are zero, the clock has been stopped low; if they are all one, its stopped high. In both of these cases the reset alarm signal `s_async_reset` goes high, otherwise it stays low.

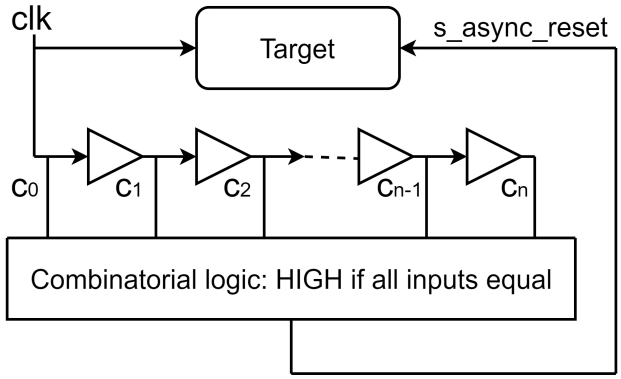

Figure 2. Asynchronous delay chain clock-stop detector circuit

c_0	c_1	...	c_{n-1}	c_n	`s_async_reset`
0	0	0	0	0	1
0	0	0	0	1	0
...	...	...	...	...	0
1	1	1	1	0	0
1	1	1	1	1	1

TABLE 1. COMBINATORIAL LOGIC ELEMENT TRUTH TABLE

The overall goal of our custom module as an appendage to a target system is to trigger an asynchronous reset upon detection of a stopped incoming clock signal, otherwise allowing the target to operate as normal (asynchronous reset not asserted) without triggering false positive resets. We note that our proposed system fails safely in the event of false positives, as resets uphold the target's security while affecting its usability. In the rest of this section we describe design considerations and requirements to make this achievable.

In order to describe the technical details of this solution we first establish some terminology. We denote the i^{th} time-shifted clock signal on the delay chain as c_i and we denote the propagation delay from the original clock signal clk to

c_i as t_i. Our solution effectively concurrently samples the incoming clock signal at n different points in time from the current clock to the clock t_n seconds earlier. t_n is the time-shifted version of the clock with the longest delay from the original clock signal.

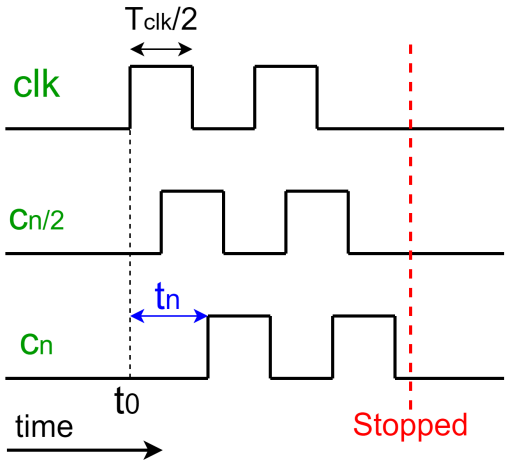

Figure 3. Timing diagram of delayed versions of clock signal

Our system samples the incoming clock at various moments in time and relies on observing variation of the clock signal as an indicator that the clock is operating normally. Therefore, the span of time across which we sample the clock must be long enough to capture its variation, otherwise the system will trigger false positives. To sample the variation of the clock signal, a key minimum bound is that $t_n > T_{clk}/2$, where T_{clk} is the nominal clock signal period. An example timing diagram with near minimum allowable t_n (where it is marginally greater than T_{clk}) is shown in Figure 3. The figure shows the temporal relationship between three different versions of the clock signal: the original clock clk, the time shifted clock c_n with the maximum delay (t_n) from the original, and one of the in-between delayed clock signals $c_{n/2}$ with approximately half of the maximum delay. At any given moment sampled in the window of time between t_0 and immediately before time 'Stopped' the values of the three versions of the clock are not all equal, which means that if these were all of the inputs to the combinatorial circuit as described in Figure 2 and Table 1 then it would not trigger an asynchronous reset, allowing the target circuit to operate as normal. 'Stopped' is the first moment where all versions of the clock (and therefore all inputs to the combinatorial circuit) are equal (at 0). This sets our combinatorial circuit output high, indicating a stopped clock and triggering an asynchronous reset of the target. If t_n were less than $T_{clk/2}$, then a false positive would always be triggered in our combinatorial circuit because all of the sample signals would be 0 immediately before the second clock pulse in clk, which is actually still a moment in time where the clock is operating normally.

t_n also sets how quickly our system can detect a stopped clock. In Figure 3 the final transition (1 to 0) of c_n occurs t_n seconds after the same transition for clk, and only after that is the stopped clock detected. While the absolute limit is that t_n must be greater than half the clock period, a larger t_n should be used in system implementation so that it is robust to clock jitter and possible variations in the chain's propagation delays. If we were to set t_n to the target's nominal clock period (e.g. $t_n = T_{clk}$), the detection time is the same as in our PLL-based solution at one clock period.

Furthermore, although digital systems are generally designed to work at a single nominal clock frequency, this proposed solution can be implemented in such a way that allows designers to specify a certain allowable frequency range of operation for the target.

Lowest Frequency Threshold. If we set t_n to a certain value then the target must be provided with a clock such that $f_{clk} > 1/(2t_n)$ which will satisfy our detection timing bound condition.

Highest Frequency Threshold. Aliasing is another potential source of false positives (failing safely). Aliasing is an effect that causes misrepresentations of discretely sampled signals due to insufficient sampling frequency which violates the Nyquist Sampling Theorem[3]. In our case, a simple example of aliasing to consider is that if our sampling frequency is exactly the same as the incoming clock frequency then each sample point is at the same phase of the clock cycle e.g. in the first half cycle where the signal is always 1, thus signal variation would not be captured.

We define the sampling frequency of our system using the shortest delay between any successive taps on the delay chain that are fed into the combinatorial logic element, i.e. $f_s = 1/(t_i - t_{i-1})$. To avoid aliasing $f_{clk} < f_s/2$.

Reset Design Constraint on Target. Our custom module imposes a constraint on the target design. Sensitive registers which receive the asynchronous reset from this circuit cannot be written to in the very first clock transition (post-idle period for gated clock signal) since the reset signal will first be de-asserted at the same time as the clock transition. This should not pose a problem because the only sequential elements that would be written to in the first clock transition in a reactivated gated system would be the input registers. Thus, the only design constraint on use of the asynchronous reset from Borrowed Time is that it must not be part of the reset logic for input registers.

4. Evaluation Setup

To evaluate our Borrowed Time countermeasure, we perform practical static power side-channel analysis attacks against a target cryptographic device, attempting to extract the secret key both with and without Borrowed Time in place. We perform a Correlation Power Analysis (CPA) [11] attack against an implementation of AES128.

We first describe a CPA attack against AES using static leakage and the metric used for assessing attacks. We then

3. $f_s > 2f_{signal}$, where f_s is the sampling frequency and f_{signal} is signal being observed.

describe the implementation we target, our measurement setup and procedure. Last, we carry out an end-to-end attack against the unprotected implementation to validate the setup.

4.1. CPA Against AES

Rijndael AES. The Advanced Encryption Standard, AES [47], is a ubiquitous symmetric block cipher with 128, 192, and 256-bit key variants that perform 10, 12, and 14 rounds of computation, respectively. Figure 4 depicts the steps of the AES encryption algorithm. Decryption of AES ciphertext uses the same steps performed in reverse, with each operation inverted. The algorithm is based on repeated rounds of the following operations:

- *AddRoundKey*. *XOR* operation with the round key and either the initial plaintext (before first round) or (thereafter) the state output from the previous round.
- *SubBytes*. Byte-wise substitution (SBox).
- *ShiftRows*. With the 16 bytes represented by a 4x4 grid, the bytes in row 1, 2, 3, and 4 undergo a circular shift by 0, 1, 2, and 3 columns to the right, respectively.
- *MixColumns*. A column-wise invertible linear transform. This operation is skipped in the final round.
- Key Expansion. Known, reversible function of the original key that generates a different key for each round.

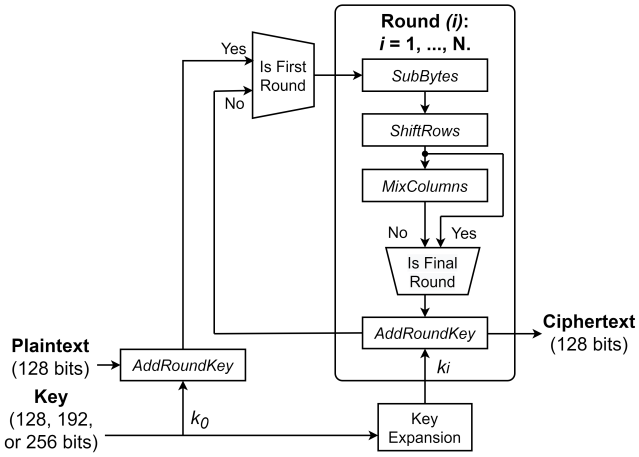

Figure 4. N-round AES encryption algorithm block diagram

Power Model. The first step for performing CPA is to select a power model. The dependency of the power consumption of a CMOS circuit on the data being processed within can be approximated by various power models. The most commonly used leakage models are Hamming weight (HW) and Hamming distance (HD) (or 'switching distance'). In the HW model [28], power consumption is dependent on the number of bits that are set on (to 1) in stored data. In the HD model [11], the consumption depends on how many bit transitions ($0 \rightarrow 1$ or $1 \rightarrow 0$) occur in a computation. Since we are concerned with static leakages which depend on fixed register contents, we use the HW power model against the intermediate, i.e. we expect that the power consumption for a given state is correlated with the number of register bits that are set to 1.

Target Intermediate. The next step is to identify a computational intermediate (a value registered at some point in execution) that depends on a combination of known values (inputs and/or outputs) with a part of the secret value that has a relatively small guess space. We opt for a final-round attack against AES128 where the target intermediate $\boldsymbol{x_i}$ is the i^{th} byte of the output state from the penultimate (ninth) round, this is also the input to the final round SBox. Our known value is the output of the encryption operation (the ciphertext). All operations in the final round of AES are byte-wise since the *MixColumns* operation is not performed. Thus, reversing the final round AES operations, each byte of the intermediate can be derived using Equation (1) from the known corresponding byte of the output ciphertext $\boldsymbol{c_j}$, and the corresponding round subkey $\boldsymbol{k_j}$ for which the guess space contains 256 candidates.

$$\boldsymbol{x_i} = \text{InverseSubBytes}(\boldsymbol{c_j} \oplus \boldsymbol{k_j}) \tag{1}$$

i is a given index of a byte of the ninth round output, and i maps to j after the final round *ShiftRows* operation.

Attack Procedure. We now describe the basic procedure of last-round CPA against AES. In the acquisition phase we perform many (N) encryptions with various random plaintext inputs, and for each encryption we take a measurement of the power consumed by the device (power trace) during a period of the computation when we know it stores the intermediate value. For a given encryption, the value of the intermediate should have an influence on that measured power trace according to our hypothesised HW power model. For each encryption we store a tuple of the power trace P and the corresponding output ciphertext $\boldsymbol{c_j}$.

Separate from acquisition is the processing phase, which we outline in Algorithm 1. For the purpose of this explanation we are concerned with one subkey byte, i.e. one i in Algorithm 1. One trace at a time, for each of the 256 possible round subkey guesses we derive the value that the intermediate would assume based on the stored output and the key guess using Equation (1), and we store the HW of this calculated intermediate (in $\boldsymbol{H}$). We end up with a $256 \times N$ matrix where each row represents the expected power consumption for all traces for a given subkey guess. The correlation between the expected power consumption for each subkey guess across all traces $\boldsymbol{H}[\boldsymbol{k_j}]$ with the measured power consumption $\boldsymbol{P}$ is then calculated. Given enough traces are used, only the correct subkey expected power consumption will exhibit correlation with the measured consumption.

For the sake of explanation we distinguished acquisition and processing phases, describing the overall process for a limited number (N) of traces. However, this does not mean we perform them one after the other, rather we perform them simultaneously. Furthermore, we do not set out to perform the attack with a predefined number of traces (N), instead during the attack we constantly evaluate the key guess correlations as more traces are acquired to see if standout candidates have emerged.

Algorithm 1 CPA against final round of AES

for $i \leftarrow 0$ to 15 **do** ▷ For all key block bytes
  $j \leftarrow \text{ShiftRowIndex}(i)$
  for *traces* $\leftarrow 0$ to N **do**
    for $\boldsymbol{k_j} \leftarrow 0$ to 255 **do** ▷ For all subkey guesses
      $\boldsymbol{x_i} \leftarrow \text{InverseSubBytes}(\boldsymbol{c_j} \oplus \boldsymbol{k_j})$
      $\boldsymbol{H}[\boldsymbol{k_j}].\text{append}(\text{HW}(\boldsymbol{x_i}))$ ▷ $\boldsymbol{H}$: $256 \times$ *traces*
    end for
    $\boldsymbol{C}[\boldsymbol{k_j}] \leftarrow correlation(\boldsymbol{H}[\boldsymbol{k_j}], \boldsymbol{P})$
    Subkey guess $\leftarrow \text{argmax}(|\boldsymbol{C}|)$
  end for
end for

Evaluation Metric. Measurements to Disclosure (MTD) is the most commonly used metric to assess the relative performance of various attack methods or parameters, and to assess the resistance of countermeasures employed by target devices. First introduced in Tiri et al. [57], MTD is the number of measurement traces taken to reveal a sought after secret value (typically a cryptographic key or subkey). More specifically in the case of CPA, MTD is defined as the cross-over point between the correct key guess correlation coefficient and the maximum of correlation coefficients for all wrong key guesses. The MTD approach is fundamentally tied to the attack method employed and also works as a metric representative of information overall leakage.

4.2. Cryptographic Target

Our target is a dedicated cryptographic device, configured with a third-party AES core [52] onto a Xilinx Kintex-7 FPGA. The algorithm implementation is not masked and is parallelised such that one AES round is performed per clock cycle, with one prior cycle for the initial *AddRoundKey* operation on the input. We purposely use an implementation without standard side-channel protections to bias in favour of the attacker and best illustrate the contrast in ease of key extraction without vs. with Borrowed Time deployed.

We perform a final-round attack as previously described in which the intermediate target is the input to the final round SBox, which, in our target implementation, is registered between the penultimate and final active clock transitions (rising edges). We refer to this period as the final clock cycle, the final ciphertext output is clocked out at the transition that terminates this cycle. To attack this implementation we stop the clock provided to the target during this cycle for an extended measurement period.

4.3. Measurement Setup and Procedure

Target Hardware. For evaluation we use the SAKURA-X [24] – also known as SASEBO-GIII – in our experiments. The board houses a *control* Xilinx 45 nm Spartan-6 FPGA, and a *target* Xilinx 28 nm Kintex-7 FPGA which we configure with the AES core. The control FPGA controls all of the target's I/O, including its supplied clock signal.

We derive the target power consumption traces by measuring the voltage across a shunt resistor placed in series with the target's voltage supply. The SAKURA-X board features easily accessible probe points across the shunt on the target voltage supply line. The target FPGA has a nominal 1.0 V supply voltage. As modified by Moradi [43] for conducting similar attacks, we replaced the board's original 10 mΩ shunt with a 1 Ω resistor that has a low temperature coefficient. This and the increased resistance improves our measurement accuracy. Since this replacement shunt increases the voltage burden, we set the supply voltage to arrive back up to 1.0 V at the target supply pin side of the shunt resistor by adjusting an on-board voltage-regulating potentiometer.

The control FPGA communicates with a PC via serial-over-USB. This USB connection supplies power to all devices on the SAKURA-X. To ensure there is no interference on the target voltage supply line from activity on the USB chip, we switch off the USB communication channel during any measurement periods and reopen it once complete.

Measurement Equipment. For acquiring power measurements of the target we connect a LeCroy AP 034 Active Differential Probe across the shunt resistor terminals and, since our signal of interest is made up of low-frequency/static components, we plug the probe into an oscilloscope with bandwidth-limited DC coupling set. This is in contrast to dynamic power analysis where AC coupling can be used. The probe operates with $\times 1$ attenuation and gain.

We use a LeCroy Waverunner 6100A oscilloscope with 8-bit vertical resolution set to its highest vertical accuracy (2 mV/div), and in its bandwidth limited (to 20 MHz) mode for greater accuracy from rejection of high frequency noise. This effectively applies a low-pass analogue filter with a cutoff frequency of 20 MHz. We set our oscilloscope to sample at 5 MS/s.

Clock Control and Measurement Interval. We control the clock signal provided to our target FPGA with the control FPGA. To perform the conventional attack, under the model where an adversary is outright stopping computation at an intermediate clock cycle, we stop the clock at the last clock cycle for our given extended measurement interval. We ensure that all of the inputs to the target are set to zero during the measurement period. We configure the control FPGA to send a trigger signal to the oscilloscope during the measurement interval.

We must stop the clock for a sufficiently long period such that the memory effect subsides before we start taking measurements, and we have a sufficiently long measurement period for intra-trace averaging to remove a significant amount of noise. We refer to the length of time from when we stop the clock to the beginning of measurement acquisition as the *time offset*, and to the time from the start to the end of the acquisition period as the *window length*. Previous works [41, 42] have indicated 20 ms to serve as a good rule of thumb for the length of time to wait for these effects to subside before taking static power measurements. In our experiments we wait 25 ms before then taking measurements over a 20 ms interval.

Temperature Control During Measurement. The most significant environmental factor affecting static leakage currents is temperature [42]. Increases in temperature lead to exponential increases in static leakage current. Therefore, we control the temperature environment of our target by placing it inside a climate chamber, the utility of which is two-fold. Primarily, we aim to stabilise any effect temperature variations have on leakage measurements by keeping the target at a constant temperature. Secondly, we want to create optimal conditions for attack performance which means we need to set a high temperature to amplify leakages. We set the climate chamber to 60°C, since the SAKURA-X board was found to sporadically power off at higher temperatures. The climate chamber we use is a Ratek H060 Hybridisation Oven which has a temperature control range from 6° above ambient to 99°C and ±0.2° stability. The majority of previous works which mount practical static power attacks use similar forms of temperature control [4, 8, 27, 38, 39, 41, 42].

We find that internal (in-chip) temperature effects also have considerable impact on leakage measurements. Performing computations generates heat within the target. To stabilise this effect we warm up the chip internally by performing the same computations at the same frequency as what would later be performed during the measurement phase. For example, if during the measurement period we perform ten encryption operations per second, we do the same during the warm up phase.

Post-Processing Temperature Control. While using a climate chamber does a reasonable job of stabilising the target's temperature, the target placed within is still subject to drifts in temperature due to non-ideal control stability, and variability in internal temperature distribution. Such temperature changes are inevitable even within an enclosed, controlled system. These effects are exhibited in measurement traces as slow drifts up and down in the power readings, whose variation far exceeds that between immediately adjacent traces which should exhibit data-dependence. Previous practical works have mitigated this source of noise with various post-processing strategies. In [42] the moving average over an 8-trace window is subtracted from the trace, effectively performing high-pass filtering. [43] interleaves each measurement trace a with measurement of the target in a deterministic baseline state which they then subtract from the immediate preceding trace. We have opted to remove all low-frequency signal components by applying a high-pass filter to the power traces taken over the measurement period. We use a digital fifth order Butterworth filter with a cutoff frequency of 0.6× the Nyquist frequency. We arrived at these characteristics via trial and error.

4.4. Validation

Our CPA attack is successful in performing full key recovery against the unprotected AES target. Thus, we validate our attack setup and procedure as a viable means of exploiting static power leakage. Figure 5 shows the correlation of each subkey candidates hypothesised power consumption across 30,000 measured traces. Each grey line represents the progression of correlation of the expected power consumption of a certain subkey guess with the actual measurement traces for various amounts of total traces. The red line represents the correlation of the correct subkey guess for that particular subkey byte. In this example, the correct subkey candidate for the 10th key byte clearly emerges after the first 800 traces. Note, this is from a single subkey.

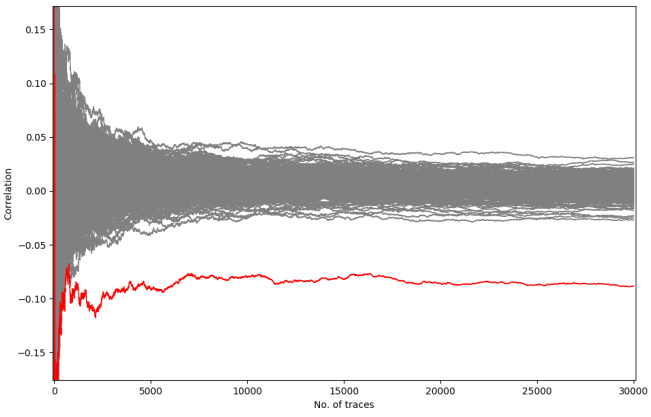

Figure 5. CPA against unprotected AES 10th key byte, 800 MTD

On average, recovering all 16 correct subkey candidates took approximately 1,500 measurement traces, hence MTD is 1,500. We calculated this average across all subkeys by using 10 disjoint subsets of our measurement trace, i.e. for each set we record the MTD for each subkey and average across 160 recoveries overall. This attack success metric provides us with a relative benchmark on the level of leakage under optimal conditions given our setup.

5. Assessing the Memory Effect.

We have described our setup and procedure for conducting a CPA attack against an AES target and validated the setup by performing a full key recovery, also producing a MTD metric indicative of our attack performance. Now, we use the same technique to assess the influence of the memory effect on overall attack performance. Specifically, we assess how attack performance is affected by the memory effect when we use acquisition windows that have shorter offsets, i.e. the window starts closer to the final active clock edge from where the clock is stopped.

We recall that due to the memory effect, dynamic effects from circuit state transitions do not subside immediately once the logical transitions themselves have terminated [44]. These effects continue to linger and have been shown to influence power measurements taken long after the transitions [41, 42, 44]. This is significant for the effectiveness of Borrowed Time as a countermeasure to static power side-channel analysis attacks because Borrowed Time drastically reduces the time that sensitive values remain leaking in a target once the clock has been stopped.

We evaluate the influence of the memory effect by repeating the attack against the unprotected target with varying time offsets and window lengths in order to gauge the duration and extent of its influence on static leakage measurements. Aside from varying these parameters, we carry out the attack as it is described in Section 4. This also extends to our analysis process, we calculate MTD by averaging across multiple independent subsets of our captured traces from the attacks. We begin by fixing the offset at 20 ms and narrowing the acquisition window length over repetitions of the attack until there is a noticeable decrease in attack performance. We find the window length of 250 µs to be the smallest that gives comparable attack performance to the 20 ms window. Performing the attack with this window length will be very sensitive to the leakage signal strength in the measurement trace.

Therefore, we then fix the window length to 250 µs and repeat the attack, shifting the time offset closer to zero each time, i.e. bringing the whole acquisition window nearer to the last active clock edge before the clock is stopped. Figure 6 shows the results of the attack for varying offsets.

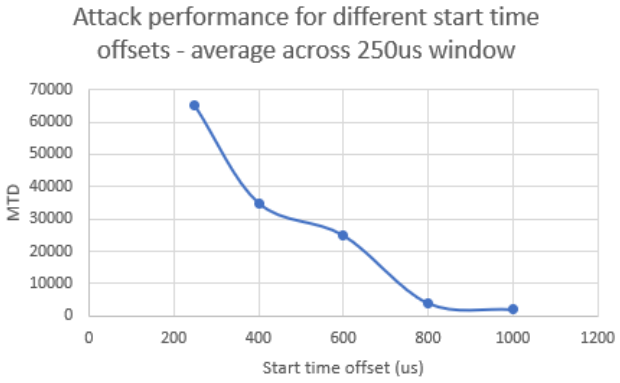

Figure 6. CPA attack performance for various start-time offsets with 250µs measurement window

We find that the memory effect appears to influence measurements for offset as late as 800 µs. Figure 7 shows an example CPA key guess progression with around 24,000 MTD at the 600 µs offset. From that point, with reducing offset the memory effect influence increases until 200 µs where the attack is unsuccessful with 100,000 measurements. We use this as our cut-off point since it is approximately two orders of magnitude greater than the optimal MTD. From these results, we deem the memory effect to last for at minimum the first 200 µs after the active clock edge.

Regarding our countermeasures, if we take this to be the memory effect duration specific to our measurement setup, any configuration that detects a stopped clock within 200 µs should protect the circuit from static power attacks. We believe that the attack is increasingly difficult for shorter offsets until a point where the memory effect noise obscures leakage to the point where the attack is infeasible. Borrowed Time (in both of its variants) can detect a stopped clock

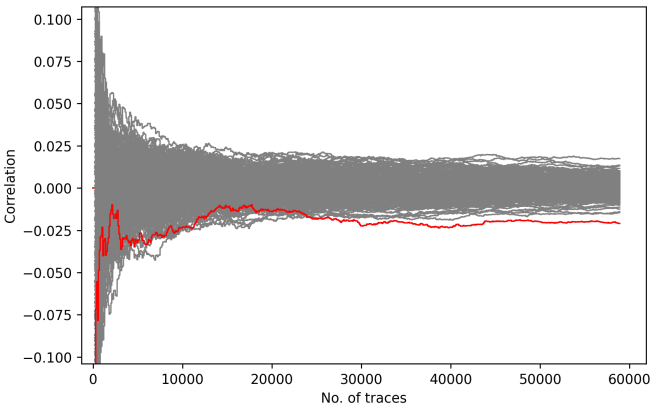

Figure 7. CPA against 10th key byte of AES with 600µs offset, 250µs measurement window, 24,000 MTD

within one clock period. This means that if Borrowed Time is deployed within a cryptographic co-processor, whose typical operating frequency would be in the megahertz range, detection and reset under a stopped clock condition occurs at latest after 1 µs.

6. Borrowed Time Implementations and Evaluation with End-to-End Attacks

Having described the general design and approach of incorporating the Borrowed Time countermeasure within a hardware system earlier in Section 3, we now describe our implementation of both of its variants within the target FPGA device. Then, using the same CPA attack evaluation setup described in Section 4 we attempt to attack the protected AES implementations.

6.1. Borrowed Time Implementations

PLL-Based Solution. For configuration onto our FPGA target, we instantiate a Xilinx MMCM [60] (Mixed-Mode Clock Manager) DCM primitive that contains a PLL which monitors the incoming clock signal. See Figure 8. This IP module is ported directly to dedicated clock management circuitry on 7-series Xilinx FPGAs. The MMCM module provides an optional status output signal from its internal PLL called `CLKINSTOPPED` which indicates that the reference input clock has stopped. This signal is asserted within one clock cycle of stoppage [60].

We verify the clock-stop detection function and the reset timing (relative to clock stop time) both through Post Place and Route Simulation performed in Xilinx's design toolchain, and on the physical FPGA target by probing an output pin tied to the reset signal.

Asynchronous Delay-Based Solution. Implementation of this variant of the Borrowed Time countermeasure on an FPGA has added complexity because we are constrained to using the resources available on the target FPGA to

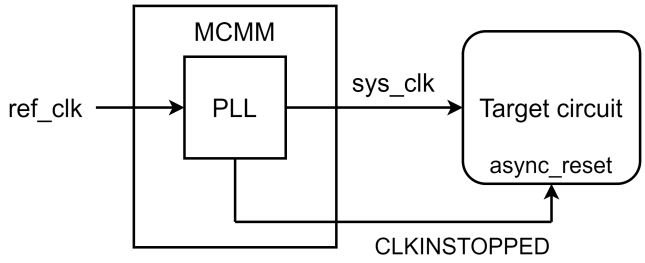

Figure 8. PLL-Based Solution Block Diagram

create a tuned delay circuit. As an aside, a corresponding ASIC implementation in which circuit elements and their placement is customisable would provide designers greater control over timing performance.

As described in Section 3, the design consists of an asynchronous system that must be tuned to a range of allowable clock operating frequencies for the target device. First, we choose a nominal clock frequency of 8 MHz for our FPGA-based AES target and design accordingly for an allowable frequency range based around that.

The next design step is to identify the digital design resources available in the platform on which the target circuit is deployed, which in our case is a 7 Series Xilinx FPGA. The most important parameter to discern is the average propagation delay of the unit delay elements that form the delay chain, since these delays define the low and high frequency threshold. To recap, the low frequency threshold is defined by the total delay between first and last elements on the chain. The high frequency threshold is defined by the minimum delay between successive taps on the delay chain.

In FPGAs, any and all combinatorial logic is implemented by lookup table (LUT) primitives. These elements are essentially configurable truth tables that can implement any Boolean function. We use LUTs to instantiate the asynchronous clock monitoring circuitry for this solution (shown in Figure 2) within our FPGA target. Our target is a Xilinx FPGA, so we design our implementation directly using the primitive LUT resource types that Xilinx build into their FPGAs [62]. Among these primitives are *LUT1* and *LUT6_2* which are 1-input 1-output and 6-input 2-output LUTs, respectively. We use *LUT1* primitives for the unit delay elements. These are effectively signal buffers. To construct the combinatorial logic element described by Table 1 we use layers of *LUT6_2* primitives.

We create a delay chain of *LUT1*s initially for the purpose of timing analysis. Post Place and Route Simulation indicates the propagation delay between successive *LUT1* elements to range from 175 to 297 ps, with approximately 200 ps on average. The delay between clk and c_0 is significantly larger at 2.7 ns due to additional routing from the clock IO buffers to configurable LUTs. Probing internal FPGA wires will not reliably show us the relative delays between taps since this would require additional routing to IO blocks which introduces significantly more delay than that between nearby combinatorial elements. This is why we use the delay value estimates provided by the design tool-chain. These estimations are reliable given they are the ground-truth used for timing analysis of digital designs. We also set design constraints to ensure that the elements are placed and routed nearby.

For a target operating frequency range of $1\,\text{MHz} < f < 12.5\,\text{MHz}$ we tune the clock monitoring circuit by taking every 200^{th} delay tap until the $10,000^{th}$ tap, giving a total delay at the final element of 20 µs. These 50 delay chain taps require three layers of LUTs making up the combinatorial logic element. The first layer is made of up of nine *LUT6_2*s that take in the delay chain taps as input. Each *LUT6_2* in the first layer has two Boolean outputs: one indicating if all inputs were equal (which we will call `EQUAL`), and the other indicating whether all inputs were 0 or 1 (`CLK_IND`). Thus, 18 inputs are input to the next layer. Each subsequent layer of *LUT6_2*s takes these input signals and passes out the same logic. Taking three (`EQUAL`, `CLK_IND`) input pairs, its output EQUAL is high if all input `EQUAL`s are high and all input `CLK_IND`s are equal. Its `CLK_IND` output is set high if all `CLK_IND`s are high. The second layer outputs six signals to the final layer *LUT6_2*.

As with the PLL-based solution, we verify the overall delay-based clock-stop detection function and the reset timing with both design simulation and physical probing of the configured FPGA.

6.2. CPA Attacks Against Protected Targets

Here we evaluate Borrowed Time in the face of our CPA attack as described and performed in Section 4.

We attack the target AES system twice, once with each variant of Borrowed Time implemented as described above. We leave all aspects of the attack setup as they are described throughout Section 4, with one necessary difference in procedure. Since Borrowed Time resets the target mid-computation upon detecting a stopped clock we do not get the correct output from attacked encryptions. As a side note, this serves as a further evidence that the deployed variants of Borrowed Time are operating correctly in terms of detection and reset alarm trigger. So that we can attain the output ciphertext needed for the attack, we repeat each random plaintext's encryption performed without clock manipulation. The extra encryptions are relatively quick and have negligible effect on attack duration.

For both variants of Borrowed Time, PLL-based (Figure 9) and delay-based (Figure 10) clock-stop detection, we find there to be no useful information leakage and are therefore unable to recover any of the subkeys with 1,000,000 traces. This is shown in the figures as the correct key guess does not emerge with higher correlation than other candidates, instead staying around 0. What this actually indicates is that there is no usable residual leakage of previous data contents following an asynchronous reset that can be exploited in the static power attack model.

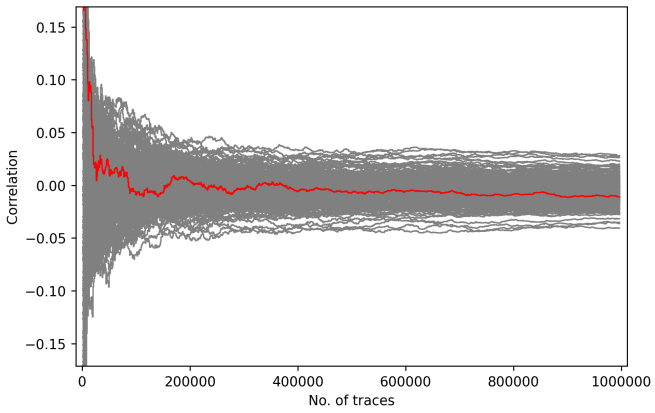

Figure 9. CPA over 1,000,000 traces against 10th key byte of AES protected with PLL-based solution, no emergence of correct candidate

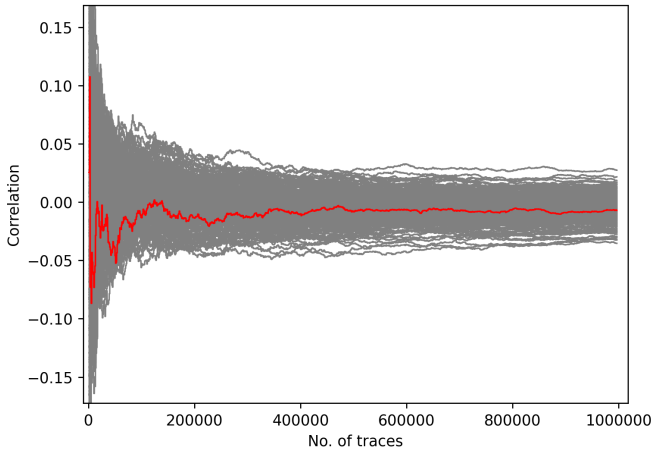

Figure 10. CPA over 1,000,000 traces against 10th key byte of AES protected with asynchronous delay-based solution, no emergence of correct candidate

7. Limitations and Future Work

A potential limitation of Borrowed Time is that while it removes sources of static leakage, we believe it may also increase dynamic leakage due to the additional switching activity of sensitive data being reset to zero. However, in the case of cryptographic systems, we note that a well-designed module clears its registers between encryptions/decryptions [48]. Hence, our countermeasure is unlikely to cause additional leakage.

Our countermeasures are likely to be ineffective against adversaries that have highly advanced invasive capability such as the capability to edit circuits, e.g. by using a focused ion beam [22]. With such a capability, an adversary can disable Borrowed Time, e.g. by disrupting the reset signal. At the same time, circuit editing allows much stronger attacks, including directly observing the contents of registers, which bypasses any need to carry out side-channel analysis. Moreover, equipment for circuit editing is quite expensive and require a high level of expertise, significantly limiting the number of potential adversaries.

Less powerful adversaries may try to disrupt Borrowed Time, e.g. by using a laser to disable the reset signal [54]. However, such fault injection attacks attacks have a low spatial resolution, laser spot diameters are on the order of at minimum 1 µm [59], whereas modern silicon features are typically below 100 nm. Hence, if system layout tightly couples the reset signal to the target circuit, isolated disruption of the reset is likely to be impractical. We leave validation of this to future work.

Resets triggered by Borrowed Time need only apply to registers that hold sensitive values at any point during execution. However, identification of sensitive registers among all others can potentially be an onerous task to expect of designers, and prone to mistakes that would defeat the purpose of implementing Borrowed Time. In such scenarios, our recommendation is to cautiously apply the reset generated by Borrowed Time to all registers within highly security critical hardware modules. As mentioned in Section 3.2, this is with the exception of the input registers of a module that is clock gated

A potential direction for further research is investigating the interaction between Borrowed Time and countermeasures that have been proven effective in preventing dynamic leakages, but have shortcomings wherein they are less effective against static leakage, such as masking [40]. We believe a complementary combination of approaches would offer a holistic security solution.

8. Conclusion

The increasing relative weight of static power in the power budget of electronic circuits may increase the risks posed by static power side-channel analysis. In this work we present Borrowed Time, a countermeasure against static power side-channel analysis attacks. Borrowed Time operates by ensuring that registers do not contain sensitive secrets when the circuit's clock is stopped. We evaluate Borrowed Time and demonstrate that it provides an effective protection against practical attacks even without any logical countermeasures.

9. Acknowledgements

We thank both Thorben Moos and Amir Moradi for helpful discussions and advice.

This work was supported by an ARC Discovery Early Career Researcher Award DE200101577; an ARC Discovery Project number DP210102670; the Defence Science and Technology Group, Australia under Agreement ID10620; and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

[1] Z. Abbas and M. Olivieri, "Impact of technology scaling on leakage power in nano-scale bulk CMOS digital standard cells," *Microelectronics Journal*, vol. 45, no. 2, pp. 179–195, 2014.

[2] D. Bellizia, G. Scotti, and A. Trifiletti, "Implementation of the PRESENT-80 block cipher and analysis of its vulnerability to side channel attacks exploiting static power," in *MIXDES*, 2016, pp. 211–216.

[3] D. Bellizia, S. Bongiovanni, P. Monsurrò, G. Scotti, and A. Trifiletti, "Univariate power analysis attacks exploiting static dissipation of nanometer CMOS VLSI circuits for cryptographic applications," *IEEE Transactions on Emerging Topics in Computing*, pp. 329–339, 2017.

[4] D. Bellizia, D. Cellucci, V. Di Stefano, G. Scotti, and A. Trifiletti, "Novel measurements setup for attacks exploiting static power using DC pico-ammeter," in *ECCTD*, 2017.

[5] D. Bellizia, M. Djukanovic, G. Scotti, and A. Trifiletti, "Template attacks exploiting static power and application to CMOS lightweight crypto-hardware," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 229–241, 2017.

[6] D. Bellizia, G. Scotti, and A. Trifiletti, "TEL logic style as a countermeasure against side-channel attacks: Secure cells library in 65nm CMOS and experimental results," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3874–3884, 2018.

[7] D. Bellizia, S. Bongiovanni, M. Olivieri, and G. Scotti, "SC-DDPL: A novel standard-cell based approach for counteracting power analysis attacks in the presence of unbalanced routing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 7, pp. 2317–2330, 2020.

[8] D. Bellizia, R. Della Sala, and G. Scotti, "SC-DDPL as a countermeasure against static power side-channel attacks," *Cryptography*, vol. 5, no. 3, 2021.

[9] D. J. Bernstein, "Cache-timing attacks on AES," 2005, preprint available at http://cr.yp.to/papers.html#cachetiming.

[10] S. Bongiovanni, F. Centurelli, G. Scotti, and A. Trifiletti, "Design and validation through a frequency-based metric of a new countermeasure to protect nanometer ICs from side-channel attacks," *Journal of Cryptographic Engineering*, vol. 5, 04 2015.

[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES*, 2004, pp. 16–29.

[12] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *USENIX Security*, 2003, pp. 1–14.

[13] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO*, 1999, pp. 398–412.

[14] M. Djukanovic, D. Bellizia, G. Scotti, and A. Trifiletti, "Multivariate analysis exploiting static power on nanoscale CMOS circuits for cryptographic applications," in *AFRICACRYPT*, 2017, pp. 79–94.

[15] B. Fadaeinia, T. Moos, and A. Moradi, "Balancing the leakage currents in nanometer CMOS logic -— a challenging goal," *Applied Sciences*, vol. 11, no. 15, 2021.

[16] T. Farheen, S. Roy, S. Tajik, and D. Forte, "A twofold clock and voltage-based detection method for laser logic state imaging attack," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 1, pp. 65–78, 2023.

[17] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *CHES*, 2001, pp. 251–261.

[18] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *J. Cryptographic Engineering*, vol. 8, no. 1, pp. 1–27, 2018.

[19] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *CRYPTO (1)*, 2014, pp. 444–461.

[20] J. Giorgetti, G. Scotti, A. Simonetti, and A. Trifiletti, "Analysis of data dependence of leakage current in CMOS cryptographic hardware," in *ACM Great Lakes Symposium on VLSI*, 2007, pp. 78–83.

[21] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. Loubet-Moundi, "High-speed ring oscillator based sensors for remote side-channel attacks on FPGAs," in *ReConFig*, 2019, pp. 1–8.

[22] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *ACM CCS*, 2013, pp. 733–744.

[23] T. Hoque, "Ring oscillator based hardware Trojan detection," Masters Thesis, University of Toledo, Ohio, USA, 2015.

[24] Y. Hori, T. Katashita, A. Sasaki, and A. Satoh, "SASEBO-GIII: A hardware security evaluation board equipped with a 28-nm FPGA," in *Consumer Electronics*, 2012, pp. 657–660.

[25] "Intel® MAX® 10 FPGA device datasheet," https://www.intel.com/content/www/us/en/docs/programmable/683794/current/pll-specifications.html, Intel, October 2022.

[26] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold leakage modeling and reduction techniques," in *ICCAD*, 2002, pp. 141–148.

[27] N. Karimi, T. Moos, and A. Moradi, "Exploring the effect of device aging on static power analysis attacks," *TCHES*, vol. 2019, no. 3, pp. 233–256, 2019.

[28] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO*, 1999, pp. 388–397.

[29] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *CRYPTO*, 1996, pp. 104–113.

[30] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *Smartcard 99*, 1999. [Online]. Available: https://www.usenix.org/

conference/usenix-workshop-smartcard-technology/design-principles-tamper-resistant-smartcard
[31] T. Krachenfels, F. Ganji, A. Moradi, S. Tajik, and J.-P. Seifert, "Real-world snapshots vs. theory: Questioning the t-probing security model," in *IEEE SP*, 2021, pp. 1955–1971.
[32] J. Krämer, D. Nedospasov, A. Schlösser, and J.-P. Seifert, "Differential photonic emission analysis," in *COSADE*, 2013, pp. 1–16.
[33] L. Lin and W. Burleson, "Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems," in *ISCAS*, 2008, pp. 252–255.
[34] P. Luo and Y. Fei, "Faulty clock detection for crypto circuits against differential fault analysis attack," IACR Cryptology ePrint Archive. 2016/967, 2016.
[35] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
[36] A. L. Masle and W. Luk, "Detecting power attacks on reconfigurable hardware," *FPL*, pp. 14–19, 2012.
[37] G. E. Moore, "Cramming more components onto integrated circuits," in *Electronics*, vol. 38, no. 8, 1965.
[38] T. Moos, "Static power SCA of sub-100 nm CMOS ASICs and the insecurity of masking schemes in low-noise environments," *TCHES*, vol. 2019, no. 3, pp. 202–232, 2019.
[39] ——, "Unrolled cryptography on silicon: A physical security analysis," *TCHES*, vol. 2020, no. 4, p. 416–442, 2020.
[40] T. Moos and A. Moradi, "Countermeasures against static power attacks: – comparing exhaustive logic balancing and other protection schemes in 28 nm CMOS," *TCHES*, vol. 2021, no. 3, pp. 780—805, 2021.
[41] T. Moos, A. Moradi, and B. Richter, "Static power side-channel analysis of a threshold implementation prototype chip," in *DATE*, 2017, pp. 1324–1329.
[42] ——, "Static power side-channel analysis - an investigation of measurement factors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 376–389, 2020.
[43] A. Moradi, "Side-channel leakage through static power – should we care about in practice?" in *CHES 2014*, 2014, pp. 562–579.
[44] A. Moradi and O. Mischke, "On the simplicity of converting leakages from multivariate to univariate," in *CHES*, 2013, pp. 1–20.
[45] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in *ISLPED*, 2001, pp. 195–200.
[46] K. Nawaz, D. Kamel, F.-X. Standaert, and D. Flandre, "Scaling trends for dual-rail logic styles against side-channel attacks: A case-study," in *COSADE*, 04 2017.
[47] "Announcing the advanced encryption standard (AES) federal information processing standards publication 197," https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, NIST, November 2001.
[48] "AES HWIP technical specification," https://opentitan.org/book/hw/ip/aes/index.html, OpenTitan.
[49] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constraints," in *CHES*, 2005, pp. 172–186.
[50] S. M. D. Pozo, F.-X. Standaert, D. Kamel, and A. Moradi, "Side-channel attacks from static power: When should we care?" in *DATE*, 2015, pp. 145–150.
[51] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*, 2001, pp. 200–210.
[52] "sasebo_giii_aes," https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html, Satoh Lab, 2016.
[53] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES - photonic side channel analysis for the rest of us," in *CHES*, 2012, pp. 41–57.
[54] S. Tajik, H. Lohrke, F. Ganji, J.-P. Seifert, and C. Boit, "Laser fault attack on physically unclonable functions," in *FDTC*, 2015, pp. 85–96.
[55] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *European Solid-State Circuits Conference*, 2002, pp. 403–406.
[56] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *DATE*, 2004, p. 10246.
[57] K. Tiri, D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "Prototype IC with WDDL and differential routing - DPA resistance assessment," in *CHES*, 2005, pp. 354–365.
[58] E. Tromer, D. A. Osvik, and A. Shamir, "Efficient cache attacks on AES, and countermeasures," *J. Cryptology*, vol. 23, no. 1, pp. 37–71, 2010.
[59] J. G. van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in *FDTC*, 2011, pp. 91–99.
[60] "7 series FPGAs clocking resources user guide (UG472)," https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking, Xilinx, July 2018.
[61] "Kintex-7 FPGAs data sheet: DC and AC switching characteristics (DS182)," https://docs.xilinx.com/v/u/en-US/ds182_Kintex_7_Data_Sheet, Xilinx, March 2021.
[62] "UltraScale architecture libraries guide (UG974)," https://docs.xilinx.com/r/en-US/ug974-vivado-ultrascale-libraries/Introduction, Xilinx, October 2022.
[63] J. Xue, T. Li, Y. Deng, and Z. Yu, "Full-chip leakage analysis for 65nm CMOS technology and beyond," *Integration, The VLSI Journal*, vol. 43, no. 4, pp. 353–364, 2010.
[64] W. Yu and S. Köse, "Security-adaptive voltage conversion as a lightweight countermeasure against LPA attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 7, pp. 2183–2187, 2017.

[65] ——, "False key-controlled aggressive voltage scaling: A countermeasure against LPA attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 12, pp. 2149–2153, 2017.
[66] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE SP*, 2018, pp. 229–244.
[67] Y. Zhou, X. Peng, L. Hou, P. Wan, and P. Lin, "Clock gating - a power optimization technique for smart card," in *ICSICT*, 2014, pp. 1–3.
[68] N. Zhu, Y. Zhou, and H. Liu, "Counteracting leakage power analysis attack using random ring oscillators," in *SNS & PCS*, 2013, pp. 74–77.
[69] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *FPGA*, 2013, pp. 101–104.