IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 463–489. DOI:10.46586/tches.v2022.i4.463-489

Redshift: Manipulating Signal Propagation Delay via Continuous-Wave Lasers

Kohei Yamashita[1], Benjamin Cyr[2], Kevin Fu[2],
Wayne Burleson[3] and Takeshi Sugawara[1]

[1] The University of Electro-Communications, Tokyo, Japan, {yamashita,sugawara}@uec.ac.jp
[2] University of Michigan, Ann Arbor, MI, USA, {bencyr,kevinfu}@umich.edu
[3] University of Massachusetts, Amherst, MA, USA, burleson@umass.edu

Abstract. We propose a new laser injection attack Redshift that manipulates signal propagation delay, allowing for precise control of oscillator frequencies and other behaviors in delay-sensitive circuits. The target circuits have a significant sensitivity to light, and a low-power continuous-wave laser, similar to a laser pointer, is sufficient for the attack. This is in contrast to previous fault injection attacks that use high-powered laser pulses to flip digital bits. This significantly reduces the cost of the attack and extends the range of possible attackers. Moreover, the attack potentially evades sensor-based countermeasures configured for conventional pulse lasers. To demonstrate Redshift, we target ring-oscillator and arbiter PUFs that are used in cryptographic applications. By precisely controlling signal propagation delays within these circuits, an attacker can control the output of a PUF to perform a state-recovery attack and reveal a secret key. We finally discuss the physical causality of the attack and potential countermeasures.

Keywords: Laser Fault Injection · Physically Unclonable Function · Delay-Sensitive Circuits · Oscillator

1 Introduction

There is a continuous demand for network-enabled embedded devices to extend ever-growing information technologies further into the physical world. Unfortunately, any new technology always comes with new attack surfaces; such embedded devices are exposed to local attackers with physical access. Ensuring security against local attackers is a challenging task because they can use physical attacks including side-channel attacks [MOP07] and fault-injection attacks [DFM+11, JT12, KSV13]. The attacks pose realistic threats to otherwise secure embedded devices, such as smartcards, and researchers have been studying new attacks and countermeasures for more than two decades.

Laser fault injection (LFI) induces faults on a target chip by applying laser stimulation [SA02]. Among many ways to inject faults [DFM+11, JT12, KSV13], LFI is considered particularly effective because of its high spatial selectivity. By illuminating a particular coordinate with a tiny laser spot, an attacker can induce precise faults such as a single bit flip in a particular address in memory. This is in contrast to the other fault-injection methods, such as clock glitching, that globally affect a target chip. The industry considers LFI a realistic threat and certification schemes (e.g., EAL5+ in Common Criteria [Joi20]) require penetration testing against LFI. To support this ecosystem, several vendors sell LFI instruments for security assessment [Risb, Alpa].

The conventional LFI focuses on digital circuits that implement cryptographic algorithms [DBC+18]. The state-of-the-art LFI setup is optimized for the peak laser power

Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

needed for a successful bitflip in digital circuits. Such a modern LFI setup is extremely expensive, typically costing more than $100,000, and is only available to well-funded attackers.

Besides LFI, there are several light-induced interferences in the wild. Xenon Death Flash [Upt15] is an issue found in Raspberry Pi 2, wherein illuminating the circuit board with a camera flash causes the system to reboot. The light interfered with a voltage regulator in a bare-silicon package and caused the problem. Another example is Light Commands [SCR+20] that silently injects voice commands to MEMS microphones with a low-power, modulated laser. The researchers identified that an ASIC inside the microphone package is one of the causes. The laser light reached the ASIC through the microphone's acoustic port and induced an electrical signal representing false audio accepted by the computer system as authentic audio.

The interesting gap between the above two attacks motivated our study. On the one hand, a conventional LFI needs an optimized high-power and short-pulse laser. On the other hand, ordinary camera flashes and laser pointers were sufficient to cause interference. The gap led us to the hypothesis that certain analog and timing circuits are more sensitive to light because they handle more variation in voltage and delay than digital circuits. If this hypothesis is correct, more light-sensitive targets opens another direction of low-cost laser injection attacks [SA02, SH07, GGS17] that extend the potential attackers from well-funded organizations to individuals with low-cost equipment.

Among many analog circuits, we focus on the ones using signal propagation delay, namely delay-sensitive circuits. They are common in cryptographic modules for realizing non-digital features using logic gates only, e.g., physically unclonable functions (PUFs) [Mae13], random-number generators [MM09], and on-chip sensors [HBB+16]. In particular, we set the ring-oscillator PUF (RO-PUF) and the arbiter PUF (A-PUF) as our targets, suspecting that laser injection would circumvent implicit assumptions of some PUF threat models.

1.1 Contributions

Manipulating Propagation Delay via Continuous-Wave Lasers (Sections 3 and 4) We propose Redshift, an attack that manipulates the behaviors of delay-sensitive circuits through laser stimulation. The targeted delay-sensitive circuits are highly sensitive to light, and Redshift works with a continuous-wave laser similar to a laser pointer, which is too weak to attack digital components. Because of this, a simple laser module in conjunction with a simple microscope is enough to perform Redshift.

Oscillator Frequency Shift (Section 5) Because of its affects on propagation delay, Redshift almost linearly controls the frequency of oscillators with increasing laser power. By applying the same laser stimulation to ring oscillators used in the RO-PUF, an attacker can almost monotonically decrease the Hamming weights in PUF states. This phenomenon is verified in a controlled manner with custom RO-PUF circuits fabricated with 180-nm and 45-nm technologies. To further indicate the feasibility of Redshift on real devices, we replicate the the light-induced oscillator manipulation on three off-the-shelf microcontrollers from different vendors: Microchip SAM L11, STMicroelectronics STM32F4, and NXP LPC55S69 [Sem21].

Arbiter PUF Manipulation (Section 6) The A-PUF, another delay-based PUF with different measurement principle, has a similar light sensitivity, too. By applying laser stimulation, an attacker can increase the propagation delay within the electrical paths of A-PUF and monotonically decrease the Hamming weights of the A-PUF states. This

manipulation is also verified with custom A-PUF circuits fabricated with 180-nm and 45-nm technologies.

State-Recovery Attack (Section 7) By exploiting the PUF states after light-injection manipulation, an attacker can recover a secret PUF state with a practical level of computational complexity. Using a combination of Redshift and an extension of Zeitouni et al.'s algorithm [ZOW+16], we successfully recover the secret PUF states in all four custom targets (180-nm RO-PUF, 40-nm RO-PUF, 180-nm A-PUF, and 40-nm A-PUF).

Causality and Countermeasures (Section 8) The conventional photoelectric model, which abstracts LFI to a current source, can explain the behaviors of the RO-PUFs and A-PUFs under laser stimulation. Finally, we suggest several countermeasures against Redshift including an improved on-chip sensor that integrates incident optical power over time.

2 Preliminary

We briefly summarize the conventional works on LFI, PUF, and its attack.

2.1 Laser Fault Injection

Semiconductor circuits are inherently sensitive to incident light energy, which can cause soft errors similar to those generated by ionizing radiation [Hab65]. Skorobogatov and Anderson first exploited such light-induced errors to attack smartcards and microcontrollers [SA02]. The attack using a laser, i.e., LFI, has several advantages over other fault-injection methods such as clock and voltage glitching. In particular, LFI enables a more precise and stealthy attack by selectively illuminating a particular coordinate. Consequently, many research works began to thoroughly investigate LFI and its applications to many different circuits [KSV13, DFM+11]. In the meantime, the industry has established the ecosystem for evaluating and certifying the resistance against LFI [Risb, Alpa, Joi20].

A parasitic photodiode explains the physical causality behind LFI [MFS+18]. Without an electrical field on the gate terminal, a MOS transistor prevents a current flowing between the source and drain terminals. A reverse PN junction between the substrate and the highly doped regions contributes to this electrical isolation, and this PN junction acts as a parasitic photodiode under laser stimulation. When laser light reaches the junction, it generates electron-hole pairs by the photoelectric effect. The generation of these carriers in the presence of the built-in electric field causes a current flow between otherwise non-conducting transistor terminals. If this photocurrent disturbs a voltage signal to an intermediate level, it can result in a bitflip. Further details about the causality are discussed in Section 8.1.

Since digital circuits periodically refresh their electrical states at clock edges, the attacker needs to inject enough optical energy within a clock cycle to cause a bitflip. But if too much energy is injected, it can cause heat buildup and permanent damage to the device. Therefore short, high-power laser pulses are necessary for a successful attack against these digital circuits. For example, a 1064-nm single-mode laser in Riscure's Laser Station 2 emits a pulse as narrow as 2 nanoseconds, with its peak power reaching 4.6 watts [Risb].

Short-pulse lasers are on the cutting edge of optical engineering, requiring special techniques such as Q-switching [Risa] and optical amplification [Alpb], which significantly increases the instruments' cost. Breier et al. reported the cost of around €150,000 [BJ15]. Meanwhile, van Woudenberg et al. estimated the cost ranging from $50,000 to $150,000 [vWWM11]. Similarly, the Joint Interpretation Library (JIL), a

working group organized for certifying cryptographic modules, categorizes the high-end laser station as *specialized*, rated between €10,000–200,000 [Joi20].

2.2 PUFs and their Application to Secure Key Storage

Physical Unclonable Functions (PUFs) are circuits that provide device-unique identifiers that are used in cryptographic modules [Mae13]. The key idea is to extract uniqueness from slight differences in each transistor (e.g., threshold voltage) due to manufacturing process variation. Researchers have been designing sophisticated circuits that efficiently harvest device-specific uniqueness with a variety of mechanisms. Some PUFs use digital components only and are available on FPGAs and semi-custom ASICs [GKST07]. Here, variation in signal propagation delay is frequently used to extract device-unique features using digital components.

Ring-Oscillator PUF (RO-PUF) [SD07] The RO-PUF uses oscillation frequencies of ring oscillators as a source of uniqueness. Each logic gate has a different propagation delay due to several manufacturing variations such as transistor sizes and dopant density. The RO-PUF efficiently extracts such variation using oscillators. The RO-PUF has a set of ring oscillators and uses their relative frequencies for generating a state.

Arbiter PUF (A-PUF) [LLG+05] The A-PUF also uses propagation delays in logic gates but with another circuit. In an A-PUF, a step signal is sent to two distinct electrical paths with slight differences in propagation delay due to device variation. An arbiter circuit determines the faster path, which is used as a binary output. Using cascaded selectors, it configures the electrical paths using challenge bits.

SRAM PUF [GKST07, HBF07] A 1-bit SRAM cell has two electrically stable states corresponding to the stored bit value. Once a cell moves to a stable state, it will stay there until a write operation overwrites it. When a chip is turned on, each SRAM cell starts from an unstable state and eventually converges to one of the two stable states. The destination is determined by manufacturing variation. Therefore, the SRAM PUF reads the SRAM's initial values and uses them as a PUF state [GKST07, HBF07].

2.2.1 Secure Key Storage Using PUF

A common PUF application is secure key management with the key encryption key (KEK) [Int20]. We denote a binary string from a PUF as the PUF state s. There are techniques for securely correcting errors in s, e.g., fuzzy extractor [GKST07]. As a result, PUF provides an error-free and device-unique key k_{PUF}, which stays within the chip during its lifetime. The system encapsulates a pre-shared key k using k_{PUF}:

$$c_k = \mathsf{Enc}_{k_{\mathrm{PUF}}}(k). \tag{1}$$

The system generates c_k at an enroll phase and stores it in external non-volatile memory. In each bootup, the system (i) generates k_{PUF} by calling a PUF, (ii) retrieves c_k from the non-volatile memory, and (iii) recovers k by decrypting c_k with k_{PUF}. The system finally provides a cryptographic service using k. Those keys disappear when the device is powered down, providing security against static reverse engineering attacks [TJ11, CSW16].

PUF-based key storage has several real-world applications. In particular, NXP Semiconductors' LPC55Sxx devices provide a set of APIs for realizing KEK using SRAM PUF [Sem19]. Other vendors use delay-sensitive PUFs for PUF-based key storage, e.g., A-PUF [DZ04] and Loop PUF [CDGB12, Sec22][1].

[1] Strong PUFs can be used for key storage by limiting the challenge space [Int20], and there are research works for choosing an efficient subset of challenges [RSGD16].

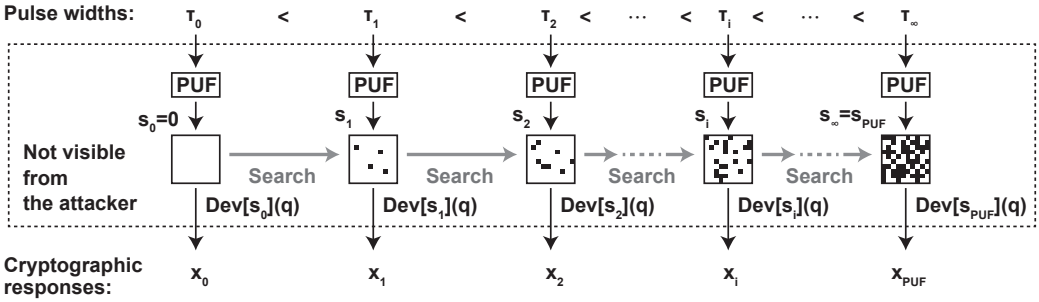

Figure 1: Illustration of Zeitouni et al.'s attack that manipulates the PUF internal state exploiting the SRAM remanence effect.

Algorithm 1 $\mathsf{Rec}(x_{\mathrm{PUF}}, \{x_i\})$

$i \leftarrow 0$; $s_0 \leftarrow 0$
repeat
  Set $i \leftarrow i + 1$
  $s_i \leftarrow \mathsf{Finder}(s_{i-1}, q, x_i)$
  if $s_i = \perp$ **then return** $\perp$
until $x_i = x_{\mathrm{PUF}}$
return s_i

Algorithm 2 $\mathsf{Finder}(s, q, x)$

Make a set of s's neighbors S
for $\hat{s} \in S$ **do**
  $\hat{x} \leftarrow \mathsf{Dev}[\hat{s}](q)$
  if $\hat{x} = x$ **then return** $\hat{s}$
end for
return $\perp$

2.3 Zeitouni et al's Attack on SRAM PUF [ZOW+16]

There are several side-channel and fault-injection attacks on PUF [Taj17]. In particular, Zeitouni et al. proposed a sophisticated attack on key storage using an SRAM PUF. The attack exploits the remanence effect: a phenomenon that an SRAM cell preserves its data for a short period after power is off [HBF07]. By exploiting the remanence effect, an attacker can partially control the PUF state.

In a normal case, the SRAM PUF generates a PUF state s_{PUF}, which is secret from the attacker. The attacker can send a query q and obtain a response $\mathsf{Dev}[s_{\mathrm{PUF}}](q)$. Here, $\mathsf{Dev}[\cdot](\cdot)$ abstracts a service using the PUF state; for example, the system recovers a pre-shared key k from s_{PUF} as described in Section 2.2.1 and returns a ciphertext obtained by encrypting q with k for challenge-and-response authentication. Here, $\mathsf{Dev}[\cdot](\cdot)$ is assumed to be public.

Figure 1 illustrates the attack process. First, the attacker writes zeroes to the target SRAM cells and resets the device for τ seconds. The normal case described above corresponds to a sufficiently long pulse, namely τ_∞. In contrast, when the pulse is very short, namely τ_0, the SRAM preserves the data across the reset by the remanence effect, and the PUF state becomes $s_0 = 0$. The attacker obtains $\mathsf{Dev}[s_0](q)$ accordingly. Then, the attacker sends a slightly longer pulse t_1, which results in an intermediate state s_1 that satisfies $\mathsf{HW}(s_0) \leq \mathsf{HW}(s_1)$ wherein $\mathsf{HW}(\cdot)$ is the Hamming weight. The attacker repeats the above process by gradually increasing the pulse widths $\tau_0 < \cdots < \tau_i < \cdots < \tau_\infty$, and the corresponding PUF states satisfy

$$0 = \mathsf{HW}(s_0) \leq \cdots \leq \mathsf{HW}(s_i) \leq \cdots \leq \mathsf{HW}(s_\infty = s_{\mathrm{PUF}}). \tag{2}$$

In the meantime, the attacker obtains the set of responses $\{x_i\}$ wherein $x_i = \mathsf{Dev}[s_i](q)$.

If the increment of the pulse widths is sufficiently small, the neighboring states are very close, i.e., $\mathsf{HW}(s_i \oplus s_{i+1})$ becomes small. In this situation, the attacker can find s_{i+1} by exhaustively searching the neighbors of s_i. By recursively repeating the process starting

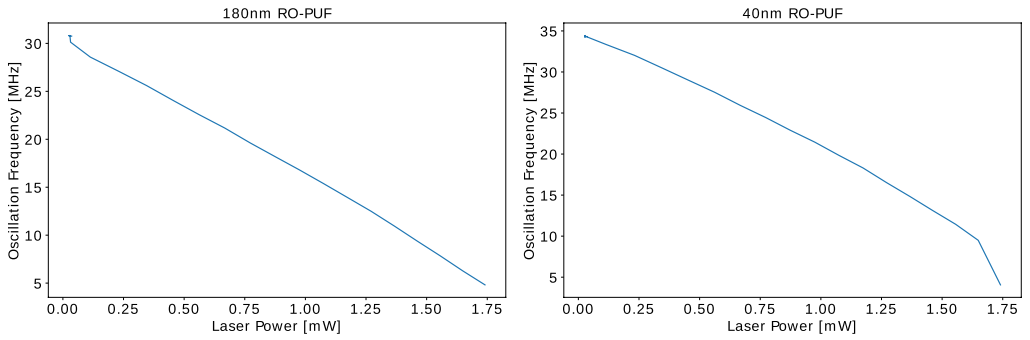

Figure 2: A ring oscillator's sensitivity to light: the oscillation frequency decreases as we increase the laser power. The results are similar across the same circuits implemented with the (left) 180-nm and (right) 40-nm CMOS technologies.

from $s_0 = 0$, the attacker eventually recovers the secret state s_{PUF}. Algorithm 1 describe the process. Here, Finder realizes the neighbor search as described in Algorithm 2; the algorithm searches for a state corresponding to the output x given the previous state s. For each neighbor $\hat{s}$, the attacker emulates Dev and obtains $\hat{x} = \mathsf{Dev}[\hat{s}](q)$. Here, $\hat{x} = x$ implies that $\hat{s}$ is the desired one. Finder returns $\perp$ when the search is unsuccessful.

3 Proposed Method

We briefly summarize Redshift and discuss its advantages, followed by the threat model describing the attacker's accessibility.

3.1 Principle

Redshift is a laser injection attack targeting delay-sensitive circuits, such as oscillators and arbiter PUFs, by changing signal propagation delay with laser stimulation. This attack is different than conventional LFIs because it changes the target's behavior within the analog domain rather than causing pure digital faults [DFM+11, JT12, KSV13]. Redshift is also different than conventional optical interferences [Upt15, SCR+20] in that it exploits spatial selectivity by using a tiny laser spot focused with a microscope.

A concrete example of Redshift is the precise control of the frequency of ring oscillators. We measure two ASIC chips fabricated with 180-nm and 40-nm CMOS technologies (Figure 2-(left) and -(right)). During the experiment, a cheap continuous-wave laser diode illuminates the target ring oscillator under a microscope. Figure 2 shows the linear relationship between the oscillation frequency (the vertical axis) and the injected laser power (the horizontal axis). The maximum laser power of 1.75 mW is several orders of magnitude lower than conventional LFI tools, and even less than the power of a laser pointer.

3.2 Advantages

Laser Injection Attack on Delay-Sensitive Circuits Redshift extends the target of laser injection attack to delay-sensitive circuits in contrast to the conventional LFIs targeting digital components for cryptography. Since delay-sensitive circuits are an essential analog building block, Redshift has many applications beyond PUFs. For example, Redshift

can degrade random number generators that use propagation delay as a source of entropy [MM09]. Delay-sensitive circuits are commonly used for on-chip sensors, too. For example, an EM sensor uses a shift in oscillation frequency to detect a magnetic-field probe for the electromagnetic side-channel attack [HHM+14]; manipulating the oscillation frequency can cause false positives and negatives in such sensors. Moreover, underclocking a system clock with a laser can result in conventional digital faults.

Stealthiness As shown in Figure 2, a low-power continuous-wave laser, too weak for conventional LFI, is sufficient for the attack. Redshift potentially evades the detection-based LFI countermeasures using on-chip sensors configured for the conventional pulse lasers. Those sensors compare the peak photocurrent with a configured detection threshold [NRV+06, MFS+18]. Hardware designers are motivated to configure the sensor with a high detection threshold [NRV+06] to avoid false positives caused by environmental lights or cosmic particles [Hab65]. The threshold is likely set low enough to sense pulse lasers but too high to sense continuous-wave lasers due to the significant difference in peak power: Redshift needs several milliwatts only, which can be less than 1/1000 of the conventional pulse lasers [Risb]. We discuss how to improve on-chip sensors to detect Redshift without sacrificing the false-positive rate in Section 8.2.

Cheaper Setup The setup for Redshift is much cheaper than the conventional laser stations [Risb, Alpa, vWWM11, BJ15]. This extends the potential attackers from well-funded organizations to individuals. In particular, this will lower the attack cost from *specialized* to *standard* in the Common Criteria certification scheme [Joi20]. Moreover, this makes the attack more stealthy in terms of the instruments' traceability because an attacker can improvise its Redshift setup using off-the-shelf components.

There are few previous works on cost-efficient optical attacks. The first LFI by Skorobogatov and Anderson [SA02] used cheap light sources such as a flashgun and a laser pointer and successfully attacked digital circuits. However, the target device was fabricated with a very old 1300-nm technology, and an attack using a laser pointer has become challenging as semiconductor chips become smaller and faster [GGS17]. As a result, recent works mostly use short-pulse lasers, as discussed previously. We empirically verified that our continuous-wave setup cannot flip digital bits with a preliminary experiment[2].

By following the above direction, Schmidt and Hutter [SH07] proposed to deliver laser light using an optical fiber instead of a microscope. Later, Guillen et al. used a flashgun combined with a single-lens optics [GGS17] to even eliminate a laser. These attacks can be even cheaper than Redshift because they do not require a microscope. These previous works aimed at achieving a high peak power using a cheap setup. In contrast, Redshift approaches the same problem by finding more light-sensitive targets. These two approaches are complementary and further optimizing the Redshift's cost using the previous techniques is open for further research. Meanwhile, the cost reduction using the previous approaches, cheaper optics and/or incoherent light source, comes at the cost of a larger spot size that makes the attack less stealthy.

3.3 Threat Model

Similar to conventional LFI, Redshift assumes a local attacker who can physically access the target chip and apply laser stimulation. Our attacks on PUFs additionally follow the model by Zeitouni [ZOW+16]: the target chip has PUF-based key storage and provides a

[2]Using the setup in Section 4, we scan SRAMs in our 180 and 40 nm ASIC chips with a continuous-wave laser using the ×20 lens and 6.3 mW laser power. We monitor the the stored values in the SRAMs during the scan, and observe no bit flip.

Algorithm 3 LIE: Getting a device response while shining a laser

Require: Laser current j and query q
Ensure: Response x
1: Set the Laser current to j
2: Invoke PUF state generation ▷ the PUF state becomes s
3: Get a response $x \leftarrow \mathsf{Dev}[s](q)$
4: **return** x

cryptographic service using a pre-shared key to which the attacker can send a query. The attacker aims to recover the secret protected by the PUF.

The attacker measures the target device with the laser-injection experiment LIE in Algorithm 3. First, the attacker keeps illuminating the target chip with laser power specified by a diode current j (line #1) while the PUF generates a state s (line #2). Finally, the attacker is able to send a query q to the chip's legitimate interface and retrieve $x \leftarrow \mathsf{Dev}[s](q)$ at line #3. Here, $\mathsf{Dev}[s](q)$ abstracts the chip's cryptographic service as discussed in Section 2.3. Here, the attacker is assumed to know the details about $\mathsf{Dev}[\cdot](\cdot)$ in the same as the previous attack by Zeitouni et al.

The availability of $\mathsf{Dev}[\cdot](\cdot)$ follows Kerckhoff's principle, and PUF-based key storage is designed to be secure without hiding it. There are open hardware for PUF wherein the assumption is reasonable [Tri17]. Meanwhile, $\mathsf{Dev}[\cdot](\cdot)$ can be unavailable in commercial chips. For example, NXP LPC55S69 uses a proprietary scheme for its SRAM-based key generation [Sem19]. In this case, the attacker should pay the cost of reverse-engineering $\mathsf{Dev}[\cdot](\cdot)$ in advance. We note that considering an attacker with reverse-engineering capability would be reasonable because protection against reverse engineering is a significant benefit of PUF-based key storage [Mae13].

4 Experimental Setup

This section provides the experimental setup and measurement procedure used throughout this paper.

Optics We use a low-cost laser module in Figure 3 composed of a laser diode, a collimation lens, and a C-mount adapter in the optical cage system. The idea is to cheaply upgrade a simple microscope, categorized as *standard* [Joi20], by attaching the laser module to the standard C-mount camera port. The total cost of the module is less than $500. The module uses a 520-nm green laser diode in the standard TO56 package that can emit up to 110 mW (Osram PLT5 520B [Osr21]), available at less than $50 from a popular online electronics retailer. We use a Wraymer RM-5400T microscope with a manual XY stage we had in our laboratory, which was roughly $4,000 at purchase. Figure 4-(left) shows the laser module installed in our microscope.

Managing the Laser Power The block diagram in Figure 4-(right) shows the components and connections used to control the laser power in a programmable way. We manage the laser power with a Thorlabs LDC202C laser driver that regulates the laser diode current. We first characterize the relationship between the diode current and the emitted optical power (the I-L curve) to translate an amount of current to laser power. For the preliminary characterization, we measure the optical power with a laser power meter (Thorlabs PM100D with the S121C sensing head) under the objective lens. The DC current output of the laser driver is controlled by a DC voltage from a function generator (Rigol DG1022Z). During our experiments, the laser power is controlled through the function

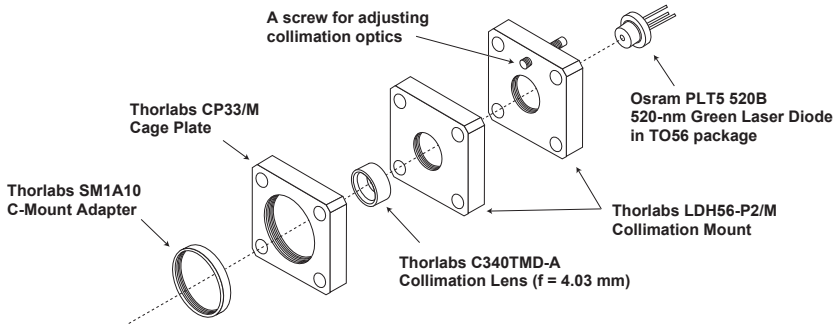

Figure 3: The laser module compatible with microscopes' standard camera port composed of a laser diode, a collimation lens, and a C-mount adapter.

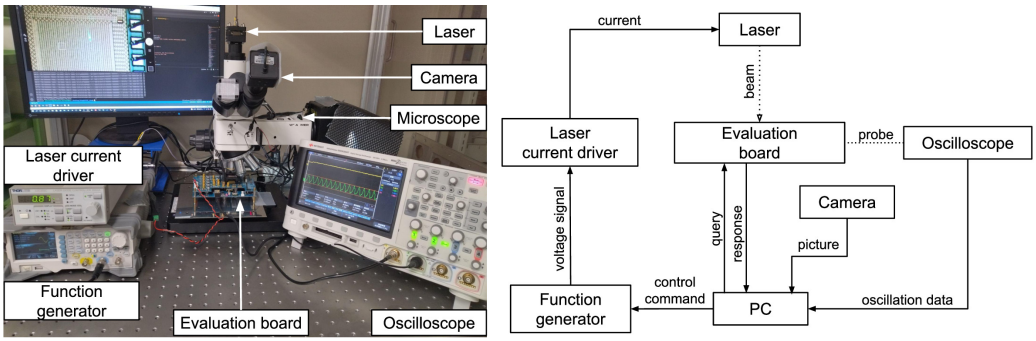

Figure 4: Experimental setup: (left) the picture and (right) its block diagram

generator's programming interface. Note that we only operate the laser diode in the linear region, i.e., beyond the threshold current.

Focusing and Magnification In general, a higher magnification is advantageous in attacking the target with smaller laser power. That is because the power density in the laser spot increases with the magnification ratio. As a drawback, however, a higher magnification requires more precise aiming. Moreover, increasing the current by a minimum unit can cause too much change in the target with too much magnification. Considering the above trade-off, we use a minimum magnification needed to cause sufficient change with the laser power around 5 mW, the power of a laser pointer. After determining an objective lens, we minimize the laser spot by changing the distance between the diode and the collimation lens using the screw shown in Figure 3. We use a camera on the microscope during the adjustment; see Figures 6 and 8 for the microscope images with laser spots. The laser spot size is proportional to the magnification ratio: the spot diameters are 14.9, 7.7, and 3.9 micrometers with the $\times 5$, $\times 10$, and $\times 20$ lenses, respectively. These spot sizes are much larger than the top metal wires in the target chips, and all our experiments succeeded without intentionally widening the laser spot. Doubling a magnification quadruples the optical energy received at a light-sensitive region covered by the spot. The laser beam does not go through an eyepiece, and its magnification does not affect the results.

Targets We evaluate Redshift on both custom ASIC chips and off-the-shelf microcontrollers. The first set of targets are RO-PUFs and A-PUFs on ASIC chips fabricated with 180-nm and 40-nm CMOS technologies. We use custom chips to perform a white-box analysis, since we know the implementation details about our RO-PUFs and A-PUFs

(shown in Sections 5 and 6). We use two chips of each PUF to compare the effectiveness of laser injection in different fabrication technologies, as they have functionally-equivalent circuits but different feature sizes. Our setup illuminates a semiconductor die from the top; we access the die by removing a glued top cover from a ceramic package. The light reaches the transistors after passing through the top metal layers, as there are 4 and 6 metal layers in the 180-nm and 40-nm chips, respectively. The chips can communicate with a PC through evaluation boards. The RO-PUF has an analog debug port that directly outputs the oscillating waveform, where an oscilloscope (Keysight DSO3034T) monitors the signal during the experiments.

The second set of targets are the clock oscillators on off-the-shelf microcontrollers. We use the three microcontrollers from different vendors available on the NewAE UFO target boards [Inc19a, Inc19b, Inc18]: NXP LPC55S69 (Cortex-M33) [Sem21], Microchip SAM L11 (Cortex-M23), and STMicroelectronics STM32F4 (Cortex M4). These devices are used to perform a black-box analysis and verify the feasibility of Redshift on real devices. The laser is injected from the top of the device after decapsulating the quad flat packages; we outsourced the decapsulation for roughly $200 per chip. The target chips are configured to output their clock signals to a GPIO pin. We directly use the 16-MHz internal RC oscillators with SAM L11 and STM32F4. For LPC55S69, on the other hand, we use a 12-MHz signal generated by dividing its 192-MHz free-running oscillator (FRO). Similar to the ASICs, we monitor the GPIO pins with the oscilloscope during the experiments.

Measurement Algorithm 4 shows the experimental procedure of repeating a unit measurement LIE (see Algorithm 3) while changing the laser power. We first fix an arbitrary query q (line #1) and gets a legitimate response x_{PUF} without laser illumination (line #2). Then, we start applying laser stimulation with the diode current starting from $j_{\min}$ to $j_{\max}$ at the step of j_{step}. As a result, we examine $j_i = j_{\min} + i \times j_{\mathrm{step}}$ for $i \in \mathcal{I} = \{0, 1, \cdots, \lceil \frac{j_{\max} - j_{\min}}{j_{\mathrm{step}}} \rceil\}$. For each current value j_i, we repeat the same measurement $r_{\max}$ times, i.e., for $r \in \mathcal{R} = \{0, 1, \cdots, r_{\max} - 1\}$ (line #5–7). We denote the r-th measurement using the laser current j_i by x_i^r (line #6). The algorithm finally returns the list of faulty PUF responses $[x_i^r]$ for $i \in \mathcal{I}$ and $r \in \mathcal{R}$ (line #11). We choose the following parameters unless otherwise noted:

- $j_{\min} = 34$ mA: the laser diode's threshold current,
- $j_{\mathrm{step}} = 0.02$ mA: the laser driver's accuracy limit, i.e., $\pm$ 0.01 mA,
- $r_{\max} = 25$: a sufficiently large number for a decimated experiment in Section 7.2.

5 Experiment: Oscillator and RO-PUFs

To show the effectiveness of Redshift, we start by controlling the delay within oscillators. First, we manipulate the frequency of a ring oscillator with laser stimulation. Second, we evaluate the same oscillator as an RO-PUF and show that we can manipulate the PUF states. Third, we replicate the frequency manipulation to microcontrollers.

5.1 RO-PUF Design

The target RO-PUF comprises of many independent oscillators, two counters, and an arithmetic comparator, as shown in Figure 5. Each ring oscillator is composed of two inverters and one NAND gate. The current source on the top roughly specifies the oscillation frequency [SD07], which we configure to be around 30 MHz. Figure 6 shows the 180-nm and 40-nm RO-PUFs with laser spots. We directly measure the oscillation using the oscilloscope during the experiment, as discussed in Section 4.

Algorithm 4 Measuring the target device while changing the laser power

Require: The minimum current $j_{\min}$, the maximum current $j_{\max}$, the current step j_{step}, and the number of iteration $r_{\max}$

Ensure: A query q, the true PUF response x_{PUF}, and the list of faulty responses $[x_i^r]$

Fix an arbitrary device query q
$x_{\text{PUF}} \leftarrow \mathsf{LIE}(0, q)$ ▷ A response with no laser injection
$i \leftarrow 0$, $j_0 \leftarrow j_{\min}$
while $j_i \leq j_{\max}$ **do**
  for $r \leftarrow 0$ to $r_{\max} - 1$ **do** ▷ Repeat the same measurement r_{max} times
    $x_i^r \leftarrow \mathsf{LIE}(j_i, q)$ ▷ A unit measurement described in Algorithm 3
  end for
  $i \leftarrow i + 1$
  $j_i \leftarrow j_{i-1} + j_{\text{step}}$
end while
return q, x_{PUF}, and $[x_i^r]$

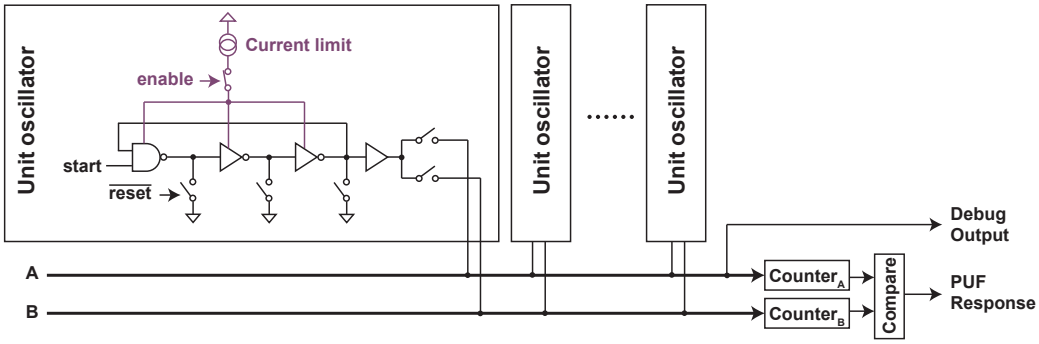

Figure 5: The target RO-PUF composed of several oscillators, two counters, and a comparator.

The RO-PUF compares the frequencies of two oscillators and generates a 1-bit state for each pair. The circuit measures the frequencies using counters that detect the number of edges as the signal oscillates. We use one reference oscillator RO_{ref} and 256 target oscillators RO_i for $i \in \{0, 1, \cdots 255\}$. We assume that the PUF generates the i-th bit by

$$b_i = \begin{cases} 0 & \text{if} \quad f_{\text{freq}}(\mathsf{RO}_{\text{ref}}) < f_{\text{freq}}(\mathsf{RO}_i) \\ 1 & \text{Otherwise} \end{cases}, \tag{3}$$

wherein $f_{\text{freq}}(\mathsf{RO})$ represents RO's oscillation frequency. Finally, the RO-PUF generates a 256-bit secret state by concatenating these bits, i.e., $s = b_0||b_1||\cdots||b_{255}$. This state s is directly output by the target chip.

5.2 Experiment 1: Changing Oscillator Frequency with LFI

First, we examine how laser stimulation affects the oscillation frequency. We locate a light-sensitive region by scanning the chip surface with a manual XY stage while monitoring RO_{ref}'s frequency. After locating a coordinate that indicates light sensitivity, we gradually increase the laser power until the oscillator stops working, i.e., observing a flat line.

Figure 2 shows the relationship between the injected laser power (the horizontal axis) and the oscillation frequency (the vertical axis). Figure 2-(left) and -(right) show the results with the 180-nm and 40-nm chips, respectively. The results clearly show the linearity

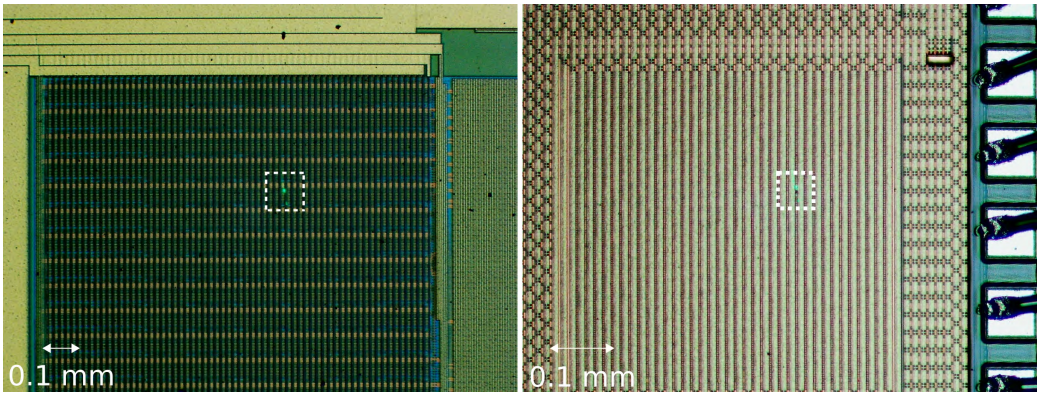

Figure 6: The microscope images of the RO-PUFs on the 180-nm (left) and 40-nm (right) chips while shining a laser. Many identical oscillators are placed in an array and the laser is focused on to illuminate a particular one.

between the frequency and the injected optical power. In the 180-nm oscillator, a ×5 magnification is sufficient to decrease the frequency from 30.7 MHz to 4.8 MHz with only 1.7 mW. The 40-nm oscillator is less sensitive because of the smaller dimensions and more metal layers, so the ×5 objective lens was insufficient. After increasing the magnification to ×10, however, the frequency changes from 34.3 to 4.1 MHz with the same laser power of 1.7 mW.

5.3 Experiment 2: Changing RO-PUF's Secret State with LFI

The frequency shift by laser stimulation impacts the bias between 0 and 1 in RO-PUF's state. We denote the oscillation frequency by f and its probability distribution by $\mathsf{Prob}[f]$. Then, the expected Hamming weight of the state s is $256 \times \mathsf{Prob}[f_{\mathrm{freq}}(\mathsf{RO}_{\mathrm{ref}}) \geq f]$. The reference oscillator's frequency is close to the median and $\mathsf{HW}(s) \approx 128$ without injection. As $f_{\mathrm{freq}}(\mathsf{RO}_{\mathrm{ref}})$ decreases, $\mathsf{Prob}[f_{\mathrm{freq}}(\mathsf{RO}_{\mathrm{ref}}) \geq f]$ also decreases, and $\mathsf{HW}(s)$ approaches 0.

Figure 7 summarizes the relationship between $\mathsf{HW}(s)$ (the vertical axis) and the power of the laser aimed at $\mathsf{RO}_{\mathrm{ref}}$ (the horizontal axis). The results clearly show that $\mathsf{HW}(s)$ approaches 0 as we increase the laser power. $\mathsf{HW}(s)$ became zero using 0.3 mW with the ×5 lens for the 180-nm chip and 0.6 mW with the ×10 lens for the 40-nm chip. At this point, $\mathsf{RO}_{\mathrm{ref}}$'s frequency is lower than any of RO_i. These laser powers are significantly smaller than the power limits that cause the oscillator to fail. By using this relationship between optical power and the Hamming weights, we can recover the PUF state s_{PUF} as shown in Section 7.

We note that the attacker can locate $\mathsf{RO}_{\mathrm{ref}}$, without the analog debug port nor the PUF output s, by repeatedly invoking Algorithm 3 while scanning the chip with a laser. If we observe several different outputs $x = \mathsf{Dev}[s](q)$ affected by the laser power, it is likely the coordinate for $\mathsf{RO}_{\mathrm{ref}}$. The attacker can distinguish it from a laser on any other oscillator RO_i, which causes at most 1-bit error in s.

5.4 Experiment 3: Clock Oscillators on Microcontrollers

To show how Redshift affects real devices, we evaluate the on-chip clock oscillators on the three microcontrollers discussed in Section 4. Like the ring-oscillator experiment, we evaluate the light-frequency characteristics by monitoring the clock signals on a GPIO pin while changing the laser power. Figure 8 (a)–(c) shows the target chips' top views with

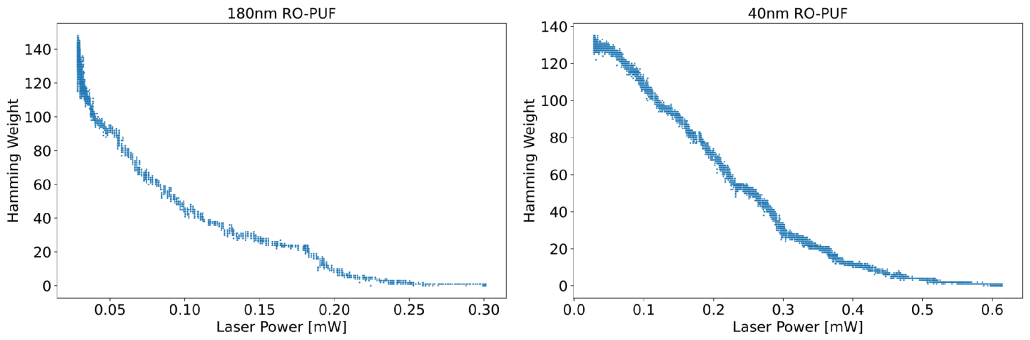

Figure 7: RO-PUF's sensitivity to light: the Hamming weight in a 256-bit PUF state $\mathsf{HW}(s)$ decreases as the injected laser power increases. The results from the 180-nm and 40-nm chips.

laser spots. Then, we obtain the light-frequency characteristics with the same procedure in Section 5.2. The ×5 objective lens was sufficient for the SAM L11 and STM32F4. Meanwhile, the LPC55S69 was significantly less sensitive, and the ×20 lens was necessary.

Figure 8 (d)–(f) are the light-frequency characteristics, which show the frequency decrease similar to the results on the custom ring oscillators. These results verify that the light-induced frequency shift is a common phenomenon that can affect many different devices. The results also show that Redshift still works with modern chips, e.g., LPC55S69 released in 2019.

The SAM L11 and STM32 are more sensitive than our 180-nm RO-PUF; less than 0.12 mW is sufficient to shift 16 to 5 MHz with the ×5 lens. The light-frequency graphs of these chips are relatively non-linear, but they are still monotonic. In contrast, LPC55S69 is much less sensitive, and 6.5 mW is necessary for shifting 12.0 to 9.1 MHz even with the highest magnification (×20). Even though it is less sensitive, its light-frequency graph is highly linear, allowing for precise oscillator control. Although explaining the exact reason for this less sensitivity needs details about the internal design, the rectangular metal patches on the top layer can be one reason. To evade the patches, we put a laser spot on a narrow space between them (see Figure 8), potentially preventing us from aiming the laser at the optimal coordinate.

6 Experiment: Arbiter PUF

We also verify Redshift on arbiter PUFs, which is another delay-based PUF with different measurement principle.

6.1 A-PUF Design

Our A-PUF circuit in Figure 9 compares the slight difference in propagation delay between two configurable delay paths. The paths have 128 stages and accepts a 128-bit challenge. Each stage is composed of a pair of selectors that changes the path depending on a challenge bit. The arbiter decides a faster path using a NAND-based SR latch; Figure 9 shows the arbiter's transistor-level internal structure for the later discussion in Section 8.1.

We use the A-PUF as a weak PUF generating a 256-bit state. We first determine random 256 challenges, namely $w_i \in \{0,1\}^{128}$ for $i \in \{0,1,\cdots,255\}$. Then, we get 256 bits by feeding these challenges to the A-PUF. The concatenated 256-bit word is the secret state s. Similar to our RO-PUF, the chip directly outputs s.

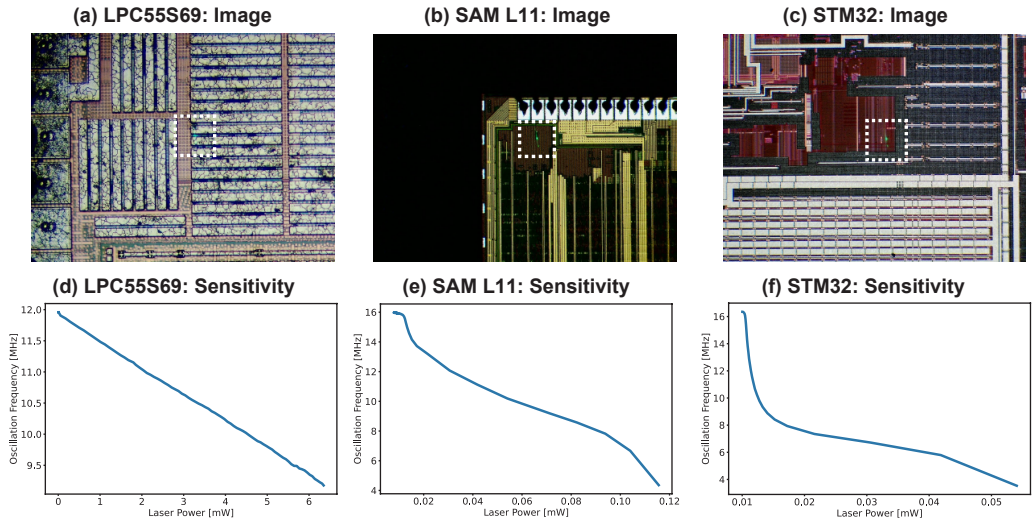

Figure 8: Light sensitivity of the clock oscillators on the microcontrollers. (a)–(c): the target chips with laser spots highlighted with the white rectangles. (d)–(f): the relationship between the oscillation frequency and laser power.

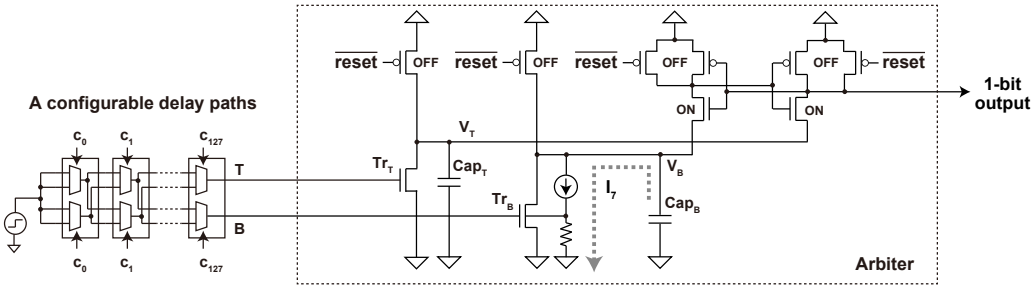

Figure 9: Our A-PUF Design. The arbiter has transistor-level description with a laser-induced photocurrent, which will be discussed in Section 8.1.

6.2 Experiment 4: Changing A-PUF's Secret State with LFI

We repeat the experiments from the previous section, only now targeting our A-PUF. We first explore the light-sensitive region by scanning the chip surface while monitoring $\mathsf{HW}(s)$. We locate two light-sensitive regions in the arbiter: one increases and another decreases the Hamming weight $\mathsf{HW}(s)$. We aim the laser beam at the former coordinate and gradually increase the laser power while measuring the corresponding $\mathsf{HW}(s)$.

Figure 10 shows the relationship between the laser power and the Hamming weight $\mathsf{HW}(s)$ while applying laser stimulation on the arbiter circuit. Figure 10-(left) and -(right) show the results with the 180-nm and 40-nm chips, respectively. Similar to the previous experiment with RO-PUF, $\mathsf{HW}(s)$ almost monotonically decreases as we increase the laser power. We use the $\times 5$ objective lens in both the 180-nm and the 40-nm A-PUFs. The 180-nm A-PUF is highly sensitive; 0.3 mW is sufficient to achieve $\mathsf{HW}(s) = 0$. Similar to the previous experiment, the 40-nm A-PUF is less sensitive. However, the 40-nm A-PUF reaches $\mathsf{HW}(s) = 0$ with 4.6 mW using the $\times 5$ lens.

The above results show that we can manipulate the A-PUF state through laser stimulation in the same way as the RO-PUF. Redshift is applicable to another delay-sensitive

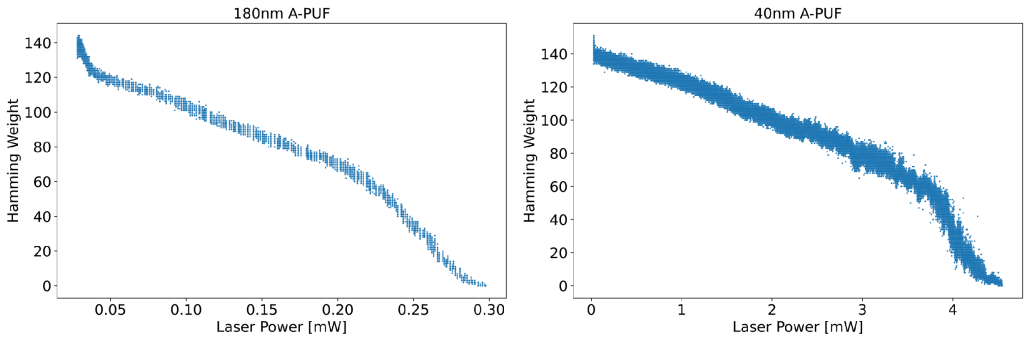

Figure 10: A-PUF's sensitivity to light: the Hamming weight in a 256-bit PUF state $\mathsf{HW}(s)$ decreases as the injected laser power increases. The results from the 180-nm and 40-nm chips.

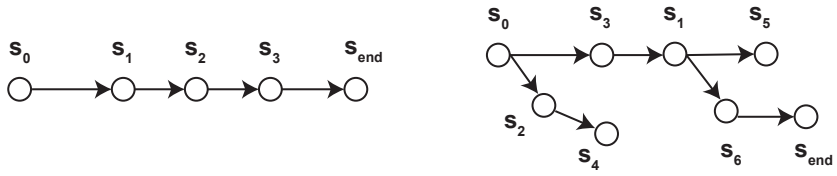

Figure 11: (Left) conventional setting without any error wherein we can always find s_{i+1} by searching s_i's neighbors [ZOW+16]. (Right) Our setting with measurement errors. The order is inconsistent and there are branches and deadends.

circuit that does not use an oscillator.

7 State Recovery Attack

We recover the PUF's secret state s_{PUF} using the data collected in our experiments. For analysis, we extend Zeitouni et al.'s attack [ZOW+16] to handle unstable bits in our PUFs.

7.1 Extension of Zeitouni et al.'s Search Algorithm

Around 5–10% of the bits in our RO-PUFs and A-PUFs are unstable (see Appendix A), which is a problem for the previous state recovery algorithm (Algorithm 1). Figure 11-(left) and (right) illustrate the state recovery with and without unstable bits. In the figure, $s_0, \cdots, s_{\mathrm{end}}$ are the intermediate PUF states. The arrow between two states means that one is reachable from another by neighbor search. Algorithm 1 successfully reconstructs each state s_{i+1} from previous state s_i when measurements are stable (Figure 11-(left)), but Algorithm 1 fails with unstable measurements (Figure 11-(right)) for two reasons. First, $\mathsf{HW}(s_i) \geq \mathsf{HW}(s_{i+1})$ is not always true when there are measurement errors. Second, there are unhandled branches and deadends (e.g., s_2, s_4, and s_5) within the search space.

To address these issues, we instead use Algorithm 5, which is an extension of Algorithm 1. Algorithm 5 takes the measured data (q, x_{PUF}, and $[x_i^r]$ from Algorithm 4) and the maximum distance $d_{\max}$ in neighbor search, and returns the secret PUF state s_{PUF} corresponding to the correct PUF response x_{PUF}. The key idea is to compare the emulated output x with a set of responses X (line #11), instead of a particular response x_{i-1} in Algorithm 1. The algorithm initializes X with all the measured responses $[x_i^r]$ (line #1)

Algorithm 5 State recovery algorithm

Require: The list of the faulty PUF responses $[x_i^r]$, the correct PUF response x_{PUF}, the query q, and the maximum search distance $d_{\max}$.
Ensure: The secret PUF state s_{PUF} or a symbol indicating failure $\perp$

1: $X \leftarrow \{x_i^r \mid i \in \mathcal{I} \text{ and } r \in \mathcal{R}\}$ ▷ Remove duplicates
2: $C \leftarrow \{0\}$
3: **while** $C \neq \emptyset$ **do**
4: $\quad C_{\min} \leftarrow \arg\min_{s \in C} \mathsf{HW}(s)$ ▷ $C_{\min} \subseteq C$ is a set of the smallest candidates
5: $\quad$ **for** $s \in C_{\min}$ **do**
6: $\quad\quad C \leftarrow C \setminus \{s\}$
7: $\quad\quad$ **for** any s' satisfying $\mathsf{HD}(s, s') \leq d_{\max}$ and $\mathsf{HW}(s) < \mathsf{HW}(s')$ **do**
8: $\quad\quad\quad x \leftarrow \mathsf{Dev}[s'](q)$ ▷ Emulate a device response for the hypothetical state s'
9: $\quad\quad\quad$ **if** $x = x_{\mathrm{PUF}}$ **then**
10: $\quad\quad\quad\quad$ **return** s' ▷ We found the answer: $s' = s_{\mathrm{PUF}}$
11: $\quad\quad\quad$ **else if** $x \in X$ **then**
12: $\quad\quad\quad\quad C \leftarrow C \cup \{s'\}$
13: $\quad\quad\quad\quad X \leftarrow X \setminus \{x\}$ ▷ The state corresponding to x is found.
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: **end while**
18: **return** $\perp$ ▷ Search failed.

and removes a particular element $x \in X$ if the corresponding state is found (line #13). We use another set C to keep track of the discovered states.

In each iteration, the algorithm first fixes a base state s and exhaustively checks its neighbors within the distance $d_{\max}$ (line #7). For each candidate s', the algorithm emulates Dev and obtains $x = \mathsf{Dev}[s'](q)$ (line # 8), which is then compared with the elements in X (line #11). If x is found in X, i.e., $\mathsf{Dev}[s'](q) \in X$, we add s' to C and remove x from X (lines # 12 and 13). After the neighbor search, the algorithm continues by choosing a new base from C. We prioritize the candidate in C with the lowest Hamming weights (line #4). If the distance between the neighboring states is closer than $d_{\max}$, we will eventually reach the final state s_{PUF} corresponding to x_{PUF}; otherwise, the algorithm returns a failure.

7.2 State Recovery Experiment

We apply Algorithm 5 to the PUF responses we obtained in Sections 5 and 6 for recovering the 256-bit PUF states s_{PUF} with the most simple Dev given by

$$\mathsf{Dev}[s](q) = s. \tag{4}$$

We discuss a more elaborate Dev in Section 7.3. Although Eq. 4 does not reflect reality, it is sufficient for evaluating the computational effort, i.e., the number of Dev emulations (line #8). Note that since the PUF output s is supposedly secret in Algorithm 5, we use it only for checking hypothetical states.

We implemented the search program with C++ and ran it on a mid-range CPU (AMD Ryzen5 2600). Table 1 summarizes the results:

- The number of unique states observed after the entire measurement $\#X$,
- The number of states #States examined for the neighbor search,
- The minimum search distance needed for a successful attack $\hat{d}_{\max}$, and

Table 1: State-recovery result with simple Dev in Eq. 4.

Target	$\#X$	#States	$\hat{d}_{\max}$	Execution Time [sec]	$j_{\max}$ [mA]
180-nm RO-PUF	6,396	3,116	3	2,306.990	43.24
40-nm RO-PUF	4,019	1,674	1	0.164	49.00
180-nm A-PUF	8,285	2,652	1	0.369	43.16
40-nm A-PUF	79,779	44,129	1	12.172	140.50

$j_{\min} = 34.00$ mA , $j_{\text{step}} = 0.02$ mA, and $r_{\max}$=25 in all the cases.
#State is the number of base states examined in the neighbor search.
$\hat{d}_{\max}$ is the minimum $d_{\max}$ needed for a successful attack.

- The total CPU time measured using the `clock` function in the C standard library.

The table also shows $j_{\max}$, the maximum laser current needed for observing $\mathsf{HW}(s) = 0$, which is necessary for a successful attack. The algorithm finished within a minute for the 40-nm RO-PUF, 180-nm A-PUF, and 40-nm A-PUF because the measurement was dense enough and $\hat{d}_{\max} = 1$. The 180-nm RO-PUF required a larger neighbor search distance of $\hat{d}_{\max} = 3$, but it still finished within an hour.

The experimental results show that we can fully recover a secret PUF state by running Algorithm 5. Even with the most challenging case, i.e., 180-nm RO-PUF, the total number of Dev emulations is $\binom{256}{3} \times 3,116 \approx 2^{33.0}$.

The search space $\binom{256}{\hat{d}_{\max}}$ increases combinatorially with $\hat{d}_{\max}$ and quickly becomes impractical. Therefore, a successful attack requires a smaller $\hat{d}_{\max}$ with a finer measurement either by reducing j_{step} or increasing $r_{\max}$. We evaluate the impact of these parameters on $\hat{d}_{\max}$ by decimating our 40-nm RO-PUF dataset[3]. Table 2 summarizes the distance $\hat{d}_{\max}$ for various j_{step} and $r_{\max}$. $\hat{d}_{\max}$ decreases with a smaller j_{step} and a larger $r_{\max}$ as expected. With sufficiently dense measurement, we can eventually achieve $\hat{d}_{\max} = 1$ in which running Algorithm 5 is trivial even with a larger PUF state.

7.3 Error Correction and Cryptography

We verify the state recovery attack with a more elaborate $\mathsf{Dev}[\cdot](\cdot)$ that is described by Algorithm 6, which includes error correction and a cryptographic service. It uses a simple error-correction scheme with a repetition code and bit selection [DGSV15], similar to the one used by Zeitouni et al. [ZOW+16].

The algorithm receives repeated measurements of the M-bit PUF state, namely $s^0, \cdots, s^{r-1}$, which are used as a bitwise $(r, 1, r)$ repetition code [BGS+08]. Algorithm 6 first decodes the repetition code with bitwise majority voting and obtains an M-bit string s (line #1). The most stable N bits of s, extracted with the predetermined indices $l_0, \cdots, l_{N-1}$, is the error-corrected key k_{PUF} (line #2 and 3). Dev then recovers the pre-shared key k by decrypting the encrypted key c_k with k_{PUF} as discussed in Section 2.2

[3]We obtained the minimum distance in Table 2 as a reachability of a graph. For a target distance d, we construct an unweighted and undirected graph $G_d = (V, E_d)$ wherein the nodes are the measured states: $V = \{s_i^r \mid i \in \mathcal{I} \text{ and } r \in \mathcal{R}\}$. We make edges if the distance between a pair of the nodes $s, s' \in V$ is shorter than or equal to d:

$$E_d = \{(s, s') \mid \mathsf{HD}(s, s') \le d \text{ and } \mathsf{HW}(s) < \mathsf{HW}(s')\} \tag{5}$$

wherein HD represents the Hamming distance. Using a standard graph algorithm, we can check if there is a path in G_d, starting from the initial state $0 \in V$ to the final state $s_{\text{PUF}} \in V$. If there is a path, we can recover s_{PUF} by running Algorithm 5 with $d_{\max} = d$. We can find the minimum distance by repeating the above procedure by incrementing d starting from $d = 1$.

Table 2: The minimum distance $d_{\max}$ needed for a successful recovery with different measurement parameters: the granularity to change the laser power j_{step} and the number of repeated measurements $r_{\max}$. We obtained the results by decimating the data measured from the 40-nm RO-PUF.

j_{step}	$r_{\max}$																							
[mA]	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0.02	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	3	3	4
0.04	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4	6
0.06	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4	5	9
0.08	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4	5	6	7	
0.10	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	6	6	7	9
0.12	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	5	5	6	6	7	9
0.14	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	5	6	6	7	8
0.16	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	5	6	7	8	8	8	
0.18	4	4	4	4	4	4	4	4	4	4	4	4	4	5	6	6	7	7	7	7	7	7	9	9	10
0.20	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	7	8	10	10	12
0.22	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	6	7	7	8	13
0.24	4	4	4	4	4	4	5	5	5	5	6	6	6	6	6	6	7	7	7	8	8	9	10	10	11
0.26	5	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	9	9	9
0.28	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	8	8	8	8	8	9	9	10
0.30	5	5	5	5	5	5	5	5	6	6	6	6	6	7	7	7	7	7	7	7	9	9	10	12	
0.32	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	9	10	10	11	11	11	11	11	15
0.34	8	8	8	8	8	8	8	8	8	8	8	8	8	8	10	10	10	11	11	11	11	11	11	12	13
0.36	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	10	10	10	10	10	10	10	10	11	12
0.38	7	7	7	7	7	7	7	7	7	7	7	7	7	7	9	9	9	9	9	9	9	9	9	12	15
0.40	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	12	12	12	12	12	13	14	17
0.42	8	8	8	8	8	9	9	9	10	11	11	11	11	11	11	11	11	11	11	11	11	12	13	14	15
0.44	8	8	8	8	9	9	9	9	9	9	10	10	10	10	10	12	12	12	12	12	12	13	13	14	17
0.46	8	8	8	8	8	11	11	11	11	11	11	11	11	12	12	12	12	12	12	12	14	14	15	16	17
0.48	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	11	11	14	14	14
0.50	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	14	15	15	15	17	17	18	19
0.52	11	11	11	11	11	11	11	11	11	11	11	12	13	13	13	13	13	13	14	14	14	14	15	16	17
0.54	11	11	11	13	13	13	13	13	13	13	13	13	14	14	15	15	15	15	16	16	16	16	16	17	18
0.56	14	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	17	17	17	18	21
0.58	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	14	14	16	16	27
0.60	12	12	12	12	12	12	12	13	13	13	13	13	14	15	15	15	15	15	15	15	15	16	17	17	20

Table 3: State-recovery result with elaborate Dev in Algorithm 6.

Target	$\#X$	#States	$\hat{d}_{\max}$	Execution Time [sec]
180-nm RO-PUF	161	150	2	0.931
40-nm RO-PUF	127	115	1	0.022
180-nm A-PUF	304	250	1	0.039
40-nm A-PUF	1842	1754	1	0.233

(line #5). Dev finally returns a ciphertext obtained by encrypting the query q with k, assuming a challenge-and-response authentication.

We evaluate the new Dev in Algorithm 6 using the dataset captured in Sections 5 and 6. We use $r = 5$, $M = 256$, and $N = 128$ by dividing the $r_{\max} = 25$ PUF states, available for each laser power, into 5×5. This results in a new dataset with $r_{\max} = 5$ composed of the outputs from Algorithm 6. We use AES-128 as the encryption algorithm and implement it using AES-NI [Gue10][4].

Following the previous work [ZOW+16], we optimize the recovery algorithm by searching k_{PUF} instead of the raw states $s^0, \cdots, s^{r-1}$. In other words, we ignore the error correction by targeting the value after error correction. We can easily extend Algorithm 5 for the optimization by (i) setting the 128-bit k_{PUF} as the target state s and (ii) skipping the lines #1–4 in Algorithm 6 while emulating Dev during the attack.

[4]We determined the bit indices $l_0, \cdots, l_{127}$ by choosing the most stable from 100,000 PUF states obtained without laser stimulation. We verified that the resulting k_{PUF} is stable; the 20,000 k_{PUF} obtained with the same data had no error.

Algorithm 6 Dev with an error correction and a cryptographic service

Require: The raw PUF outputs $s^0 \cdots s^{r-1} \in \{0,1\}^M$, encrypted key c_k, and a query q, the indices of selected bits $l_0, \cdots, l_{N-1}$.
Ensure: A response $x \in \{0,1\}^N$
1: $s \leftarrow \mathsf{Vote}(s^0, \cdots, s^{r-1})$ ▷ Bitwise majority voting
2: **for** $i = 0, \cdots, N-1$ **do**
3: $\langle k_{\mathrm{PUF}} \rangle_i \leftarrow \langle s \rangle_{l_i}$ ▷ Bit selection: $\langle t \rangle_j$ represents the j the bit of a word t
4: **end for**
5: $k \leftarrow \mathsf{Enc}^{-1}_{k_{\mathrm{PUF}}}(c_k)$
6: $x \leftarrow \mathsf{Enc}_k(q)$
7: **return** x

Table 3 summarizes the experimental results for recovering k_{PUF}, which shows smaller $\#X$, #States, and $\hat{d}_{\max}$ compared with the previous experiment. The attack is easier because of the smaller search space reduced from 256 to 128 bits and more stables bits by the error correction. As a result, the execution times are faster even though new Dev involves two AES calls. All the searches finished within 1 second. The most challenging case is the 180-nm RO-PUF with $\hat{d}_{\max} = 2$ that finished in 0.931 seconds. Dev occupies only ≈10% of the total execution time; the AES encryption and decryption with AES-NI is faster than other utility functions for data structures.

We finally discuss how error correction impacts the state recovery attack. Error correction introduces difficulty in two ways. First, the state size before error correction is larger. Second, it extends the distance between the neighboring states. That is because Dev will not generate a different output until a sufficient amount of difference is accumulated [ZOW+16]. Searching the state after error correction, as we did in our experiment, is a general strategy to improve the attack. However, this technique becomes less efficient with a larger codeword. We consider generating an N-bit key using the κ-bit codeword N/κ times in parallel. In this case, the number of adjacent states after error correction is $2^\kappa \cdot N/\kappa$. The bitwise repetition code in our experiment is the most efficient case with $\kappa = 1$. However, the complexity grows exponentially with κ and eventually becomes infeasible as κ increases.

8 Discussion

This section includes the discussions on the causality and possible countermeasures, followed by the related works.

8.1 Causality

We investigate how laser injection influences the transistors based on the conventional parasitic-photodiode model.

Oscillators and RO-PUF Figure 12 shows an inverter within a ring oscillator; Figure 12-(left) and -(right) show the current paths during high-to-low and low-to-high transitions, respectively. The gray arrows are the legitimate current paths. Meanwhile, the dashed arrows are the additional current paths caused by laser-induced photocurrent; the laser stimulation is modeled by the addition of extra current sources.

We focus on the high-to-low transition in Figure 12-(left) for simplicity. Without a laser injection, all the PMOS current I_1 goes to the load capacitance, and thus $I_2 = I_1$. With a laser injection, part of I_1 goes to ground through the photocurrent I_3 causing $I_2 < I_1$. A smaller I_2 increases the low-to-high transition delay because the inverter needs more time

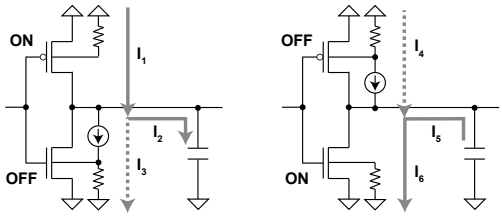

Figure 12: An inverter with the laser-induced photocurrent making (left) high-to-low and (right) low-to-high transitions during an oscillation.

charging the load capacitance with a smaller I_2. The high-to-low transition delay, shown in Figure 12-(right), increases in the same way. Finally, the longer propagation delay in each inverter results in a slower oscillation.

A-PUF Figure 9 shows the transistor-level description of the arbiter with the current sources by laser stimulation. The arbiter's frontend is a pull-down network composed of the transistors ($\mathsf{Tr_T}$ and $\mathsf{Tr_B}$) and the capacitors ($\mathsf{Cap_T}$ and $\mathsf{Cap_B}$). The capacitors are precharged by negating the reset before sending a step signal; the figure shows the transistor states after the precharge. The capacitors preserve their charges because both $\mathsf{Tr_T}$ and $\mathsf{Tr_B}$ are turned off. At the evaluation phase, a step signal propagates through the delay paths and eventually turns $\mathsf{Tr_T}$ (resp. $\mathsf{Tr_B}$) ON. Then, the capacitor $\mathsf{Cap_T}$ (resp. $\mathsf{Cap_B}$) starts to discharge through the transistors. When the capacitor voltage $\mathsf{V_T}$ (resp. $\mathsf{V_B}$) reaches a threshold, the backend cross-coupled inverters converge to a stable state representing the faster path[5].

Figure 9 shows a case a laser illuminates $\mathsf{Tr_B}$ only. The photocurrent I_7 discharges $\mathsf{Cap_B}$, making the delay in discharging $\mathsf{Cap_B}$ shorter because a part of the charges is already lost when the step signal finally arrives. This causes the arbiter a bias preferring the bottom path at B, increasing the population of corresponding bit value. Laser stimulation on $\mathsf{Tr_T}$ (cf. $\mathsf{Tr_B}$) causes an opposite bias. This explains the two light-sensitive coordinates observed in Section 6: one increases 0 and another increases 1.

Extension to Other Analog Circuits We discuss the potential to extend Redshift into other analog circuits beyond the delay-sensitive circuits that were the focus of this paper. The simple principle of the parasitic-photodiode model —laser stimulation generates a current in an otherwise insulating transistor— is still useful for explaining Redshift, as discussed above. Therefore, the model should be useful for predicting how Redshift affects a target analog circuit in advance. For an experimental verification, sweeping the laser power while monitoring an output will be generally useful. An amplitude-modulated laser will be necessary for targets that may reject DC signals, e.g., a microphone [SCR+20]. Once a problem is experimentally verified, we can use the parasitic-photodiode model in a SPICE simulation to make a quantitative analysis and evaluate the effectiveness of a countermeasure.

8.2 Countermeasures

On-Chip Sensors The conventional sensor-based countermeasures should work in principle if a detection threshold is properly configured for Redshift. However, as discussed

[5]The target A-PUF has an additional functionality to increase the load capacitance $\mathsf{Cap_T}$ and $\mathsf{Cap_B}$ for calibrating a bias caused by asymmetry in the layout design. For simplicity, we conducted our experiments by disabling this feature by disconnecting all the additional capacitors. To further eliminate the possibility of this functionality being the cause of light sensitivity, we conducted another experiment with the additional capacitors fully connected. The state recovery attack was still successful.

in Section 3.2, simply raising the detection threshold can prohibitively increase the false positives caused by environmental lights or cosmic particles [Hab65, NRV+06]. Instead, we can improve the false-positive rate by integrating (averaging) the sensors' output over time. The above technique is effective because Redshift is sustained for a longer period. We can achieve this by adding an integrator circuit after a conventional LFI sensor. Alternatively, an oscillator-based sensing scheme naturally achieves such integration. He et al. proposed to use a ring oscillator to detect laser pulses [HBB+16]; it can be extended for efficiently detecting Redshift.

Detecting a Wrong PUF Key. Detecting a wrong PUF key and terminating the cryptographic service $\mathsf{Dev}[s_i](q)$ [ZOW+16] can prevent the state-recovery attack. We can achieve this with recalculation. At enrollment, we encrypt a constant value, such as 0, to get the corresponding ciphertext $c_0 = \mathsf{Enc}_{k_{\mathrm{PUF}}}(0)$ and store it on a non-volatile memory along with the encapsulated pre-shared key c_k in Eq. 1. After recovering the PUF key k'_{PUF} on each bootup, the system recalculates $c'_0 = \mathsf{Enc}_{k'_{\mathrm{PUF}}}(0)$ and compares it with the stored c_0. An unsuccessful comparison means $k_{\mathrm{PUF}} \neq k'_{\mathrm{PUF}}$, and we can terminate the following sensitive operations.

Changing a Reference Oscillator Redshift on RO-PUFs becomes more difficult if the RO-PUF design does not contain a fixed reference oscillator. For example, Merli et al.'s chaining method [MSE10] generates PUF states by comparing adjacent oscillators:

$$b_i = \begin{cases} 0 & \text{if} \quad f_{\mathrm{freq}}(\mathsf{RO}_i) < f_{\mathrm{freq}}(\mathsf{RO}_{i+1}) \\ 1 & \text{Otherwise} \end{cases} . \tag{6}$$

To attack this scheme, the attacker should change the laser coordinate for each bit, which significantly increases the difficulty of the measurement.

Obfuscation As discussed in Section 3.3, hiding Dev with a proper hardware obfuscation scheme [FBT17] will prevent the attacker from running the state-recovery algorithm in Section 7.

8.3 Related Works

Laser-Assisted Device Alteration (LADA) [RE03, BHK13] LADA is an LSI reliability analysis for isolating a failure mechanism typically in a digital circuit. LADA injects a continuous-wave laser on a target transistor while checking the Shmoo plot, the pass/fail test with different frequencies and voltages. If the injection changes the plot, the target transistor has a small operational margin and is a potential failure cause [RE03]. Both Redshift and LADA change transistor behavior with continuous-wave laser injection. Boit et al. mentioned LADA as a modern diagnosis tool at FDTC 2013 [BHK13], but there is no concrete attack so far, as far as the authors are aware. Also, LADA usually targets logical pass/failure in a digital circuit, not in delay-sensitive circuits.

Ring Oscillator as an On-Chip Sensor for LFI [HBB+16] He et al. proposed an oscillator-based sensor for detecting laser injection on FPGA [HBB+16]. The authors discovered a laser pulse disturbed oscillation and proposed a circuit to detect them using a phase-locked loop. This sensor uses the ring oscillator's sensitivity to light. Meanwhile, this method focused on detecting laser pulses and did not cover the frequency manipulation through laser power. In the meantime, He et al.'s method can be an efficient countermeasure for detecting Redshift, as discussed in Section 8.2.

LFI of PUFs on FPGA [TLG+15] Tajik et al. studied LFI on several PUFs realized on FPGAs. The authors used a pulse laser to overwrite volatile memory storing FPGA's configuration. This causes a critical degeneration in the target PUF, such as suspended oscillation. Attacking such a degraded PUF is easy. Although the attack targets PUFs, it is still a conventional LFI attacking a digital component using a pulse laser. Also, the attack is limited to FPGAs (cf. ASIC).

Fault Sensitivity Analysis [LSG+10] Li et al. proposed fault sensitivity analysis (FSA) [LSG+10] that induces faults with various intensities, e.g., various pulse widths in clock glitching. The attacker then exploits the correlation between the fault intensity and the faulty ciphertexts. Redshift uses the same strategy; it applies laser stimulation with various laser power. Meanwhile, FSA focuses on the conventional digital faults.

Helper Data Manipulation Attack There are attacks targeting error correction schemes in PUF-based key storage by manipulating public helper data. In particular, several attacks observe if a manipulation causes a decoding failure or not [DGSV15, Bec19]. Compared with the helper data manipulation attack, Redshift directly manipulates a raw PUF output instead of helper data. As a result, Redshift bypasses several countermeasures against helper-data manipulation attacks that prevent/detect modifications in helper data [DGSV15].

Electromagnetic Fault Injection (EMFI) EMFI is a fault-injection technique that applies electromagnetic disturbance [SH07, AH20, TCG+21]. By using a tiny injection probe near the target chip, EMFI can cause localized faults. EMFI's setup can be even cheaper than Redshift and can work without chip decapsulation [AH20]. Meanwhile, EMFI and LFI have different principles and properties. In particular, EMFI's spatial resolution is generally worse [AH20, TCG+21]. Moreover, the conventional EMFI attack digital circuits only.

9 Conclusion

We proposed a new laser injection attack on delay-sensitive circuits that are highly sensitive to light. The attack is feasible by using a low-power, continuous-wave laser that significantly reduces the attack cost and is more stealthy against sensor-based countermeasures. We experimentally verified that we could manipulate the frequency of oscillators by changing the injected laser power on our custom ASIC and the off-the-shelf microcontrollers. An attacker can leverage the above phenomenon to manipulate the PUF states from our ring-oscillator PUFs. A similar state manipulation is possible on arbiter PUFs, showing that the proposed attack can be extended beyond oscillators. Our recovery algorithm, extended from Zeitouni et al.'s attack, successfully recovered secret information by exploiting the manipulated PUF states.

There are several interesting problems to explore in the future. Extending Redshift to other applications and analog circuits can be an interesting challenge. Also, the causality discussed in Section 8.1 needs further verification through circuit simulation and controlled experiments.

Acknowledgment

This work is sponsored in part by the SECOM Science and Technology Foundation and the Archimedes Center for Healthcare and Device Security.

References

[AH20] Karim M. Abdellatif and Olivier Hériveaux. Silicontoaster: A cheap and programmable EM injector for extracting secrets. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020*, pages 35–40, 2020.

[Alpa] AlphaNov. Double laser microscope station for IC security evaluation - fault injection. https://www.alphanov.com.

[Alpb] AlphaNov. Pulse-on-demand modules PDM series. https://www.alphanov.com.

[Bec19] Georg T. Becker. Robust fuzzy extractors and helper data manipulation attacks revisited: Theory versus practice. *IEEE Trans. Dependable and Secure Computing*, 16(5):783–795, 2019.

[BGS+08] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on FPGAs. In *Cryptographic Hardware and Embedded Systems, CHES 2008*, volume 5154, pages 181–197, 2008.

[BHK13] Christian Boit, Clemens Helfmeier, and Uwe Kerst. Security risks posed by modern IC debug and diagnosis tools. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2013)*, pages 3–11. IEEE Computer Society, 2013.

[BJ15] Jakub Breier and Dirmanto Jap. Testing feasibility of back-side laser fault injection on a microcontroller. In *10th Workshop on Embedded Systems Security, WESS 2015*, page 5, 2015.

[CDGB12] Zouha Cherif, Jean-Luc Danger, Sylvain Guilley, and Lilian Bossuet. An easy-to-design PUF based on a single oscillator: The Loop PUF. In *2012 15th Euromicro Conference on Digital System Design*, pages 156–162, 2012.

[CSW16] Franck Courbon, Sergei Skorobogatov, and Christopher Woods. Reverse engineering flash EEPROM memories using scanning electron microscopy. In *15th Smart Card Research and Advanced Applications - CARDIS 2016*, pages 57–72, 2016.

[DBC+18] Jean-Max Dutertre, Vincent Beroulle, Philippe Candelier, Stephan De Castro, Louis-Barthelemy Faber, Marie-Lise Flottes, Philippe Gendrier, David Hely, Regis Leveugle, Paolo Maistri, Giorgio Di Natale, Athanasios Papadimitriou, and Bruno Rouzeyre. Laser fault injection at the CMOS 28 nm technology node: an analysis of the fault model. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018*, pages 1–6, 2018.

[DFM+11] Jean-Max Dutertre, Jacques JA Fournier, Amir-Pasha Mirbaha, David Naccache, Jean-Baptiste Rigaud, Bruno Robisson, and Assia Tria. Review of fault injection mechanisms and consequences on countermeasures design. In *2011 6th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6. IEEE, 2011.

[DGSV15] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. Helper data algorithms for PUF-based key generation: Overview and analysis. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 34(6):889–902, 2015.

[DZ04] Srinivas Devadas and Thomas Ziola. Volatile device keys and applications thereof, 2004. US7839278B2.

[FBT17] Domenic Forte, Swarup Bhunia, and Mark M. Tehranipoor. *Hardware Protection through Obfuscation*. Springer Publishing, 2017.

[GGS17] Oscar M. Guillen, Michael Gruber, and Fabrizio De Santis. Low-cost setup for localized semi-invasive optical fault injection attacks - how low can we go? In *8th Constructive Side-Channel Analysis and Secure Design - COSADE 2017*, pages 207–222, 2017.

[GKST07] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *9th Cryptographic Hardware and Embedded Systems, CHES 2007*, pages 63–80, 2007.

[Gue10] Shay Gueron. White paper: Intel advanced encryption standard (AES) new instructions set revision 3.0. Available at https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf, 2010.

[Hab65] D. H. Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Tran. Nuclear Science*, 12(5):91–100, 1965.

[HBB+16] Wei He, Jakub Breier, Shivam Bhasin, Noriyuki Miura, and Makoto Nagata. Ring oscillator under laser: Potential of PLL-based countermeasure against laser fault injection. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016*, pages 102–113, 2016.

[HBF07] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *The Conference on RFID Security*, 2007.

[HHM+14] Naofumi Homma, Yu-ichi Hayashi, Noriyuki Miura, Daisuke Fujimoto, Daichi Tanaka, Makoto Nagata, and Takafumi Aoki. EM attack is non-invasive? - design methodology and validity verification of EM attack sensor. In *Cryptographic Hardware and Embedded Systems, CHES 2014*, volume 8731, pages 1–16. Springer, 2014.

[Inc18] NewAE Technology Inc. STM32Fx UFO target, 2018.

[Inc19a] NewAE Technology Inc. LPC55S69 UFO target product datasheet, 2019.

[Inc19b] NewAE Technology Inc. SAML11 UFO target, 2019.

[Int20] International Organization for Standardization (ISO). Iso/iec 20897-1: Information security, cybersecurity and privacy protection — physically unclonable functions — part 1: Security requirements, 2020.

[Joi20] Joint Interpretation Library. Application of attack potential to smartcards version 3.1. https://www.sogis.org/, 2020.

[JT12] Marc Joye and Michael Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012.

[KSV13] Dusko Karaklajic, Jörn-Marc Schmidt, and Ingrid Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 21(12):2295–2306, 2013.

[LLG+05] Daihyun Lim, Jae W. Lee, Blaise Gassend, G. Edward Suh, Marten van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225, pages 320–334, 2010.

[Mae13] Roel Maes. *Physically Unclonable Functions - Constructions, Properties and Applications*. Springer, 2013.

[MFS+18] Kohei Matsuda, Tatsuya Fujii, Natsu Shoji, Takeshi Sugawara, Kazuo Sakiyama, Yu-ichi Hayashi, Makoto Nagata, and Noriyuki Miura. A 286 F^2/cell distributed bulk-current sensor and secure flush code eraser against laser fault injection attack on cryptographic processor. *IEEE J. Solid State Circuits*, 53(11):3174–3182, 2018.

[MM09] A. Theodore Markettos and Simon W. Moore. The frequency injection attack on ring-oscillator-based true random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2009*, pages 317–331, 2009.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[MSE10] Dominik Merli, Frederic Stumpf, and Claudia Eckert. Improving the quality of ring oscillator pufs on fpgas. In *5th Workshop on Embedded Systems Security, WESS 2010*, page 9, 2010.

[NRV+06] Egas Henes Neto, Ivandro Ribeiro, Michele G. Vieira, Gilson I. Wirth, and Fernanda Lima Kastensmidt. Using bulk built-in current sensors to detect soft errors. *IEEE Micro*, 26(5):10–18, 2006.

[Osr21] Osram Opto Semiconductors GmbH. PLT5 520B green laser diode in TO56 package. https://dammedia.osram.info/media/resource/hires/osram-dam-6652476/PLT5%20520B_EN.pdf, 2021.

[RE03] J.A. Rowlette and T.M. Eiles. Critical timing analysis in microprocessors using near-IR laser assisted device alteration (LADA). In *International Test Conference, 2003. Proceedings. ITC 2003.*, volume 1, pages 264–273, 2003.

[Risa] Riscure. DPSS laser: Stable high intensity laser for green and NIR datasheet v1.1. https://getquote.riscure.com/picdb/filedb/3792/DPSS%20laser%20datasheet_1.1.pdf.

[Risb] Riscure. Laser station 2. https://www.riscure.com/product/laser-station-2/.

[RSGD16] Olivier Rioul, Patrick Solé, Sylvain Guilley, and Jean-Luc Danger. On the entropy of physically unclonable functions. In *IEEE International Symposium on Information Theory, ISIT 2016*, pages 2928–2932. IEEE, 2016.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems, CHES 2002*, pages 2–12, 2002.

[SCR+20] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light Commands: Laser-based audio injection attacks on voice-controllable systems. In *29th USENIX Security Symposium*, 2020.

[SD07] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, 2007.

[Sec22] Secure-IC. Physically unclonable function (puf) ip. Available at https://www.secure-ic.com/products/issp/security-ip/key-management/puf-ip/, 2022. Accessed: 2022-07-04.

[Sem19] NXP Semiconductors. AN12324: LPC55Sxx usage of the PUF and Hash Crypt to AES coding rev. 0. https://www.nxp.com/docs/en/application-note/AN12324.pdf, 2019.

[Sem21] NXP Semiconductors. LPC55S6x product data sheet, 2021.

[SH07] Jörn-Marc Schmidt and Michael Hutter. Optical and EM fault-attacks on CRT-based RSA: Concrete results. In *15th Austrian Workhop on Microelectronics, Austrochip 2007*, pages 61–67, 2007.

[Taj17] Shahin Tajik. *On the physical security of physically unclonable functions*. PhD thesis, Technical University of Berlin, Germany, 2017.

[TCG+21] J. Toulemont, G. Chancel, Jean Marc Gallière, Frédérick Mailly, Pascal Nouet, and Philippe Maurine. On the scaling of EMFI probes. In *18th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2021*, pages 67–73, 2021.

[TJ11] Randy Torrance and Dick James. The state-of-the-art in semiconductor reverse engineering. In *Proceedings of the 48th Design Automation Conference, DAC 2011*, pages 333–338. ACM, 2011.

[TLG+15] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser fault attack on physically unclonable functions. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2015*, pages 85–96, 2015.

[Tri17] Tribler. Open hardware-based and patent-free physical unclonable function (PUF). Available at https://github.com/Tribler/tribler/issues/3064, 2017.

[Upt15] Liz Upton. Xenon Death Flash: a free physics lesson. https://www.raspberrypi.org/blog/xenon-death-flash-a-free-physics-lesson/, 2015. Accessed: 2021-11-20.

[vWWM11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical optical fault injection on secure microcontrollers. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2011*, pages 91–99, 2011.

[ZOW+16] Shaza Zeitouni, Yossef Oren, Christian Wachsmann, Patrick Koeberl, and Ahmad-Reza Sadeghi. Remanence decay side-channel: The PUF case. *IEEE Transactions on Information Forensics and Security*, 11(6):1106–1116, 2016.

A Evaluation of Unstable Bits

This section describes how we evaluated the unstable bits in the PUF outputs. For the target bit, we consider it stable if we get the same bit value for any $r \in \mathcal{R}$; otherwise, we consider it unstable. We counted the number of unstable bits as follows. For each index i that represents the laser current, we count the number of unstable bits as

$$h(i) = \mathsf{HW}\left(\underset{r,r' \in \mathcal{R}, r \neq r'}{\mathsf{OR}} \left(\mathsf{XOR}(s_i^r, s_i^{r'})\right)\right) \tag{7}$$

wherein OR and XOR represent bitwise operations over 256-bit words. Table 4 summarizes the average and standard deviation of $h(i)$ regarding i. Table 4 summarizes the number of unstable bits in our measurements; roughly 5–10% are unstable in 256 bits.

Table 4: The average and standard deviation of the unstable bits within a 256-bit PUF state

Target	Average [bit]	Standard deviation [bit]
180-nm RO-PUF	17.02	9.85
40-nm RO-PUF	14.19	8.26
180-nm A-PUF	19.10	5.15
40-nm A-PUF	21.11	5.78