Black-Box Laser Fault Injection on a Secure Memory

Olivier Hériveaux

Ledger

Abstract. With the constant development of electronic devices, their increasing complexity and need for security, cryptography in embedded systems has become a strong requirement to protect data or secure communications. Some devices run on basic microcontrollers, which are vulnerable to low-budget physical attacks allowing the retrieval of secret materials, as shown in previous publications. More sophisticated devices use dedicated security circuits able to withstand higher levels of physical attacks. This paper describes a hardware attack conducted on the ATECC508A CryptoAuthentication secure memory, a circuit used in many security applications and IoT devices for strong authentication. In particular, it is used in the Coldcard Mk2 Bitcoin hardware wallet to securely store the seed. We present an attack using Laser Fault Injection, in a practical approach from an attacker perspective, where we retrieve the content of secret data slots in the mentioned wallet specific configuration, allowing an attacker to steel the protected funds. Contrary to security-certified chips, this circuit has a public datasheet. Nevertheless, its implementation and its firmware are proprietary, allowing only a black-box approach. Finally, we assess the difficulty of this attack in the given real-case scenario and demonstrate it is a practical attack despite the high setup cost.

1 Introduction

Most of modern systems rely on cryptography to secure communications, authenticate devices and users, or securely store information. Many devices store critical information, such as cryptographic private keys, in basic microcontrollers or memories. In a hardware context, where the attacker has physical access to the device, implementing secure software to protect sensitive data against logical attacks is necessary but not sufficient. Powerful techniques have been developed to recover secrets using non-invasive, semi-invasive or invasive attacks [13].

Side-channel analysis exploit physical leakages to recover keys during the execution of a cryptographic algorithm. Operations leakage can be observed in power measurement [5], electromagnetic radiation [10], computation time [6], silicon photonic emission [11] or even acoustic noise as shown in [3]. This field of expertise has grown in the past decades,

and sophisticated software and hardware counter-measures have been developed to prevent the exploitation of such leakages.

Invasive probing attacks can be conducted to spy circuit internal signals, which may convey secrets. Focused Ion Beam stations can also be used to directly edit a circuit by cutting metal tracks and/or creating new ones. Initially developed for failure analysis, such equipment can be diverted to disable hardware security circuitry, or routing-out internal signals for further probing or manipulation of sensitive data. This latest technique is among the most expensive. Secure-elements usually implement a top-metal shield to detect such attacks and thus make those much more difficult. But, for oldest node technologies, this can still be bypassed with a lot of time and effort, usually many months [16]. This class of attacks requires very expensive equipment (up to millions USD), a high level of expertise, and is very time consuming.

Other attacks are based on fault injection to produce exploitable computation errors in circuits. This class of attacks is usually referred as semi-invasive. Faults can be injected by different means. An attacker controlling the clock can introduce glitches on the clock signal. This usually inserts setup/hold violations at the gates level, and thus produces logical errors. Another easy way to introduce faults consists in generating glitches on the power supply. Finally, Electromagnetic Fault Injection and Laser Fault Injection are often more efficient while they require a higher level of expertise and more expensive equipments. These methods have higher temporal and spatial resolutions, enabling local and precisely targeted effects. For instance, Laser Fault Injection is very efficient to bypass security features, as many previous work described [9, 12, 14, 15, 17]. Equipment can be expensive for precise laser stations, but efficient low-cost setup can also be designed. The research time and effort for semi-invasive attacks is much smaller than invasive attacks. Some vulnerabilities can be identified within a week. The exploitation time is often reduced: from one day using Laser Fault Injection, down to a few minutes with Electromagnatic Fault Injection or power glitching.

We chose to evaluate the resistance of the ATECC508A circuit against high potential attackers, as defined by [4]. Firstly, we developed tools and dedicated electronics to communicate with the target device and be able to send commands, with proper instrumentation for power trace analysis, circuit power management and precise triggering. We prepared samples for backside illumination, eventually thinning down the silicon. We then conducted a fault injection campaign using a focused laser beam.

In this paper, we present a security evaluation of secret data storage of the ATECC508A, in the particular configuration of the Coldcard Mk2 hardware wallet. In Section 2, we introduce the target of evaluation and its security mechanisms. In Sections 3 and 4, preparation and setup are described. Eventually, the testing campaign is explained in Section 6, leveraging the information gathered from the power traces as described in Section 5. Finally, Section 7 details the results of the testing campaign.

The critical exact setup parameters of the attack are not given: the settings have been communicated to the manufacturer, and we estimate it is not useful to disclose them in this paper. It limits on the field exploitation, and gives sufficient time to the vendor to mitigate the issues and warn its customers.

2 Target of Evaluation: ATECC508

The ATECC508A is a secure memory with NIST-P256 Elliptic Curve cryptography for IoT authentication applications. It provides key generation, secure storage for keys or small data blobs, and supports cryptographic algorithms such as ECDSA, ECDH, HMAC, etc. This circuit is presented as a *Secure Element* by the Microchip (Atmel) manufacturer, but from our knowledge has no public security certification regarding its resistance against (physical) attacks.

Before the publication of this article, the complete datasheet of this circuit was publicly available [8]. For this reason, manufacturers might have chosen this circuit for its accessibility rather than contracting Non-Disclosure-Agreements required for other products. This even allows building Open-Source hardware designs using the ATECC508A circuit to protect sensitive data. Training samples can be freely acquired from many component resellers.

2.1 Coldcard Hardware Wallet

The Coldcard hardware wallet, in its Mk2 version,[1] uses this circuit as a secure storage for sensitive secrets: pairing key between the secure memory and the MCU, user PIN code and master seed. The master seed is a very critical asset: it grants the ownership of corresponding cryptoassets and allows signing transactions on the blockchain to transfer funds protected by the device. In this application the stored data is not related to P256 curve in any way, and the ATECC508A circuit is only used as a secure

1. Mk3 version has upgraded to ATECC608A

storage with authenticated access through the knowledge of the PIN code and pairing secret.

Fetching the secret seed from the secure memory is done by the microcontroller in the following (simplified) steps:

1. The microcontroller proves knowledge of the pairing secret by answering correctly to a challenge from the secure memory. Success unlocks use of the PIN hash data slot, required to perform the next two steps.
2. The PIN entered by the user is hashed together with the pairing secret: if the PIN is correct, the result matches the content of the PIN hash data slot:

   ```
   SHA256( SHA256( pairing secret + 0h58184d33 + PIN ) )
   ```

3. The seed data slot is read and decrypted using the hash as key.

Knowledge of the pairing secret and PIN hash data slots is enough for getting access to the hardware wallet seed. Optionally, the PIN code can be recovered by brute-forcing the hash. Note that the pairing secret can also be retrieved by attacking the STM32 microcontroller, which is known to be vulnerable to low-cost glitch attacks [1, 7].

2.2 Software security mechanisms

The ATECC508A circuit has an internal ROM memory for storing the proprietary firmware (unknown to us), and EEPROM memory for storing data. The EEPROM memory is used to store both configuration data (CONFIG zone) and user secret data (DATA zone). Direct access to the EEPROM memory is not possible, and the circuit firmware implements commands to configure the access rules, put and retrieve data inside defined memory slots.

The DATA zone of the EEPROM memory is split into 16 data slots. Data slots have different fixed sizes. The smallest slots store 36 bytes each, and the largest one stores 416 bytes.

Each data slot has an access configuration which is defined in the CONFIG EEPROM memory sector. A default factory configuration is defined for a typical use case. It can be changed to modify the access conditions of the data slots. Depending on the configuration, accessing to a data slot may require being authenticated, and communication during read or write commands can be encrypted. Once the configuration has been set, it must be permanently locked by executing the *Lock* command. When the configuration is locked, data slots must be provisioned and then

locked permanently. The device is operational when the CONFIG and DATA sectors are both locked.

The hardware wallet we studied stores the master seed in a 72 bytes data slot. The ATECC508A device is not capable of running the cryptographic algorithms necessary to sign transactions for the Bitcoin blockchain. Therefore, this secret is fetched by the MCU which runs all the cryptographic calculations. The ATECC508A will return the secret seed encrypted after verifying the user has knowledge of the hash of the PIN code.

The hash of the PIN code is stored in another data slot of the secure memory. This data slot can store up to 36 bytes, but only the first 32 bytes are used by the wallet. The data slot configuration has the "is secret" bit set, meaning the secure memory will return an error for any read memory attempt on this data slot.

2.3 Hardware security counter-measures

As stated in the datasheet [8], the circuit has a top-metal mesh preventing front-side probing attacks. Without such a mesh, an attacker might be able to connect to circuit internal wires using very thin needles and a probing station, and readout sensitive data during the execution of the circuit (such as the data bus). We wanted to verify that this counter-measure is present. We had little equipment for front-side preparation but we managed to observe this shield anyway (Figure 1). For this, we milled the package in front-side using a diamond milling tool. The main difficulty is to stop the milling process at the right time before touching the silicon. When the remaining plastic package was thin enough to see the circuit by transparency, we stopped the milling process and finished gently with a scalpel.

Figure 1 is a picture of the observed top-metal shield. We can see a curious labyrinth-like pattern covering the surface of the chip, which is probably made of one or multiple wires filling all the space to hide the underneath logic. The chip has been damaged by the process and is not functional anymore: scratches of the milling tool are visible on the top-left corner of the picture, and the bonding-wires have been removed.

The circuit generates its own clock source internally to avoid basic clock glitching attacks. The CPU core voltage is also regulated internally to prevent voltage glitch attacks. We verified the effectiveness of this counter-measure against direct voltage glitches applied to the external pins, or using electromagnetic pulses with a short-distance antenna.

Fig. 1. Active shield visible in front-side

There is no mention to resistance against laser fault injection (such as light detectors for instance), which is why we decided to test the device against this class of attacks.

The memory of the device is internally encrypted. This is a good counter-measure against electrical memory probing attacks, however, as we show in this paper, with the correct attack path we can rely on the circuit to decrypt the content for us during an attack.

Measuring the power traces during our experiments revealed temporal jitter during commands execution (i.e. noise on the execution time). Calling a same command twice produces slightly different power traces. This can be either natural CPU clock noise or a voluntary counter-measure. There is no mention to this mechanism in the datasheet. Like shooting an arrow on a randomly moving target, this counter-measure makes harder fault injection and the reproducibility of attacks is therefore severely degraded, especially without dedicated real-time synchronization equipment.

3 Sample preparation

To prepare the circuit for laser fault injection, the package must be opened in backside. We used an ASAP1 machine (micro-milling tool dedicated for chip decapping) (Figure 2) with a 1 mm diamond tool for milling the package. Once the copper lead frame was visible, it has been removed using a 1 mm metal milling tool. This step must be performed with extra care as milling down too deep may scratch or destroy the silicon under the lead frame. Once the lead frame has been removed, the

conductive glue paste between the silicon and the lead frame is gently removed with a scalpel and wood toothpick. The Figure 3 shows the internals of a circuit package, and how we prepared the backside access. Figure 4 shows a photo of a prepared sample.

Fig. 2. ASAP1 micro-milling machine for chip decapping

We estimated the die thickness to be around 250 μm. It has been measured optically with our microscope by focusing firstly on the surface of the silicon and secondly on the visible circuit gates. To obtain the die thickness t, the displacement difference during focusing must be multiplied by the refractive index of the silicon in infrared light, which is approximately $n_{\text{si}} = 3.5$.

$$t = \left| \frac{z_{\text{surface}} - z_{\text{gates}}}{n_{\text{si}}} \right|$$

The measurement is not very accurate, therefore we rounded it to the nearest known standard die thickness. Knowing the die thickness is required when thinning down the silicon.

We also used the microscope stage and camera to measure the chip dimensions.

During our experiments, we used a laser source for fault injection. We found out the laser can be powerful enough to inject faults without

thinning down the silicon, making sample preparation easier and with very limited risk of destruction. We also performed some tests on thinned samples, which allows an attacker using a cheaper laser source.

The chip is rotated inside its package (Figure 4), which is a bit unusual and requires particular caution for silicon substrate thinning and daughter board wiring.

Chip width	1585 μm
Chip height	1410 μm
Chip surface	2.235 mm^2
Substrate thickness	250 μm

Table 1. Physical measurements of the chip

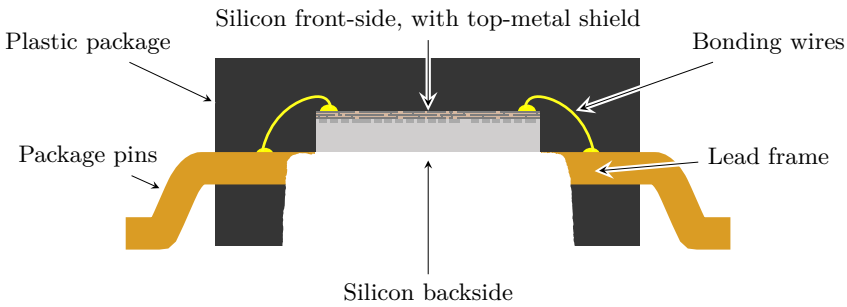

Fig. 3. SOIC package milling for backside access

Using the infrared camera mounted on the XYZ stage of our laser test-bench, we took several pictures of the silicon from backside. The images have then been stitched to produce a complete picture of the circuit, as shown in Figure 5.

The images stitching positions are the locations returned by the XYZ stage, which is accurate enough to avoid using any advanced stitching algorithm. The camera images are averaged to reduce noise and get better contrast, and a post-processing filter is applied to remove dust particles on the lens and reduce thumbnail/shadowing effect due to non-homogeneous lighting of the sample through the microscope.

In a black-box approach, we need to identify the different parts of the circuit before trying to inject faults. When available, open documentation from the manufacturer can help matching circuit blocks such as memories

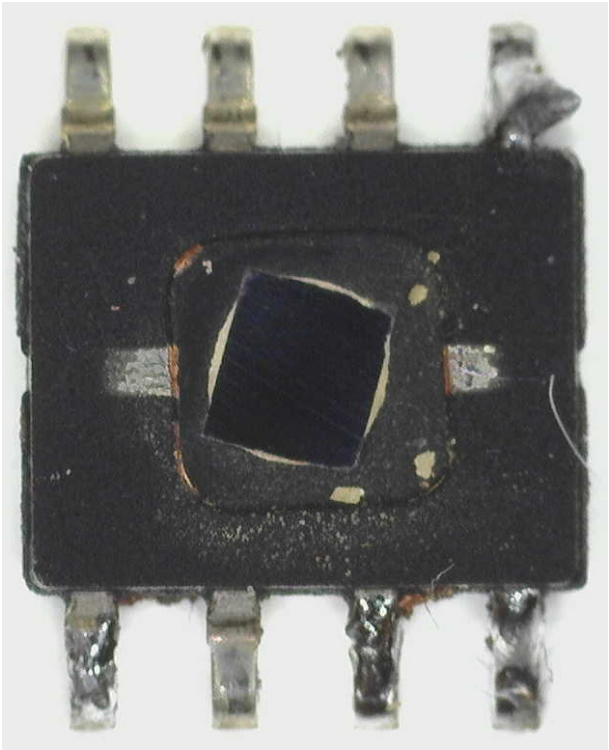

Fig. 4. ATECC508A chip in SOIC-8 package, backside

or peripherals to indicated specifications. Comparison to other circuits from the same manufacturer can also help identification.

According to the datasheet [8], the EEPROM memory stores 11200 bits. This could be verified on the EEPROM layout using our microscope camera, as shown in Figure 6. 16 banks of 700 bits are visible. Each bank is probably mapped to a word bit (horizontal bit lines). 50 columns are visible (vertical word lines), and we supposed that each column stores 14 bits.

The ROM memory of the chip has a much smaller cell size than the EEPROM memory. We were not able to count the memory cells due to our microscope limitations. Furthermore, we don't know what ROM technology is implemented, but an attacker with more equipment may try to find out if this is a contact ROM which could be extracted and reversed, provided the ROM addresses and bits are not scrambled. Knowledge of the firmware binary of this circuit would be a significant advantage in setting-up attacks and understanding the chip errors after fault injections.

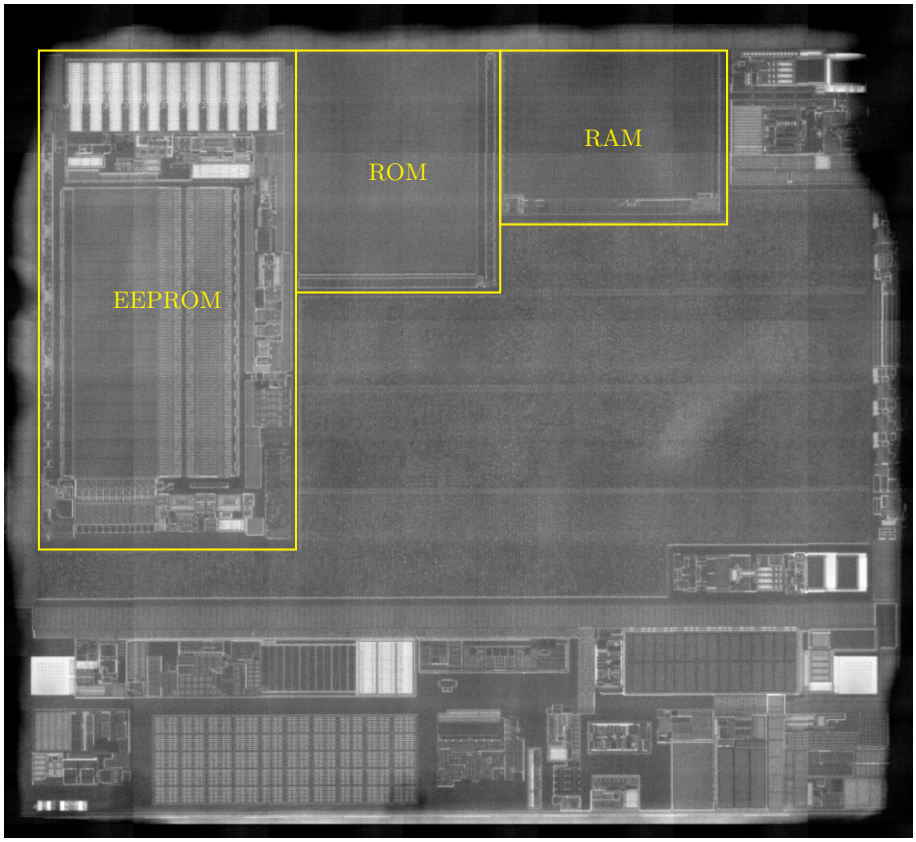

Fig. 5. Infrared backside image of the circuit and memory floorplan

4 Setup

4.1 I²C communication and triggering

The ATECC508A circuit exists in two interface versions: I²C or Single-Wire. The communication interface mode is programmed in the internal EEPROM memory during manufacturing and cannot be changed. We used the I²C version for our setup. Only four pins have to be connected: VCC, GND, SDA and SCL.

Communication with the device is performed using Ledger Donjon Scaffold board and its Python API [2]. Commands are sent to the device following the protocol defined in the datasheet [8]. An I²C write transaction is performed to execute a command, and the response is fetched with an I²C read transaction after command completion. Each response may include a

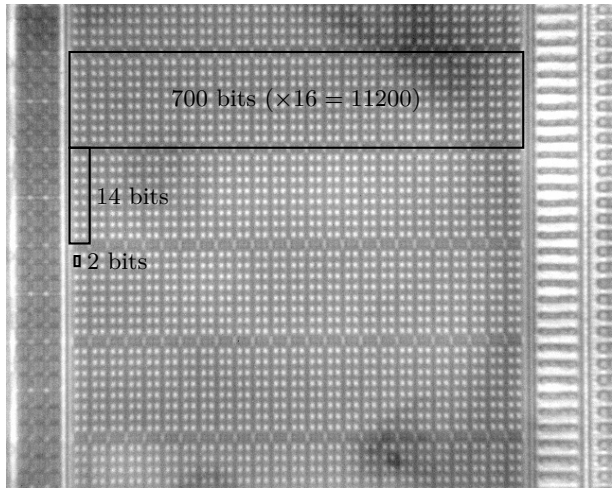

Fig. 6. EEPROM memory cells. Captured with 50X magnification objective lens

vendor-specific error code, giving useful information for diagnostic after fault injection.

The I^2C peripheral of Scaffold was used to generate accurate trigger during read and write transactions. We used this trigger source to start the Scaffold's configurable delay and pulse generator connected to the infrared laser source. With this setup we are able to inject faults synchronized with the I^2C transactions, with very low jitter.

The 3.3 V power supply of the device under test is controlled by the Scaffold board. It is switched OFF and ON before each new test to recover from possible circuit crashes.

4.2 Power measurement

The device power consumption reflects its activity. Each instruction executed by the CPU has a signature on the power trace. In a black-box approach, it is quite hard to interpret a power trace. However, comparing different power traces given different execution paths can provide an attacker good hints on the best timing for fault injection. Also, patterns and their repetitions can give interesting information.

In our setup, we measured the power consumption of the device under test using a 20 Ω shunt resistor (R1) connected between the circuit ground pin and the board ground, as shown in Figure 8. We didn't use any decoupling capacitor to prevent low-pass filtering. The signal is amplified

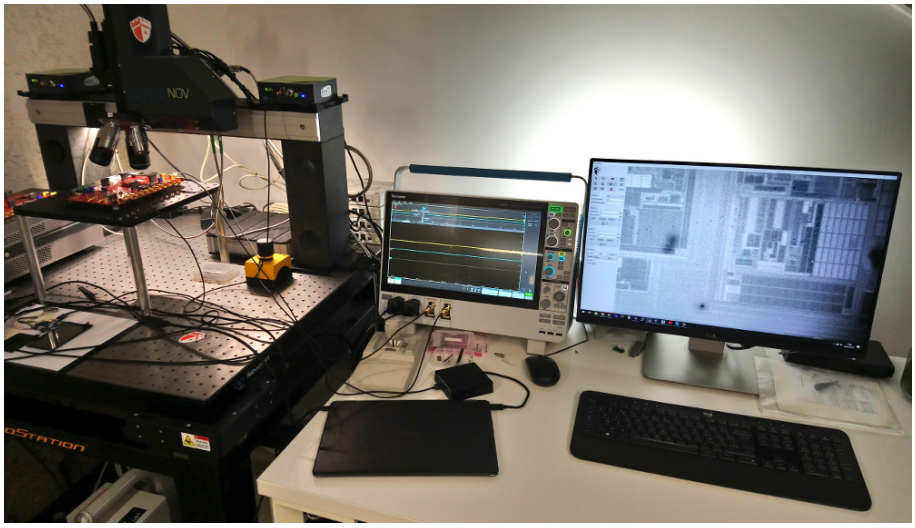

Fig. 7. Our Laser Fault Injection test bench

using an operational amplifier close to the resistor (gain $G = 1 + \frac{R2}{R3} = 11$). The output of the amplifier is then fed to an oscilloscope which records the power traces.

4.3 Targeted asset and attack path

Prior to the fault injection campaign, we programmed the devices in the same exact configuration as the Coldcard wallet. This configuration is detailed in Table 2. We loaded the PIN hash data slot with easily recognizable data: `0123456789abcdef...` It is important to note that the data slots cannot be written if the configuration is not locked, and the circuit will then operate normally only when both configuration and data zones are locked. There is no possibility to rollback to factory settings, therefore any will to change the configuration or data for testing purpose will require using another sample.

The "is secret" flag of the data slot is set: read is strictly prohibited. This data slot can only be used to prove to the secure memory the knowledge of the PIN code and unlock other data slots.

In black-box approach, it is hard to identify quickly possible attack paths, as the implementation details and protections of the functionalities are totally unknown. We had to make an attack path hypothesis and then try out. Furthermore, it is highly recommended to choose a path where only one fault is enough to bypass the security. Performing multi-fault

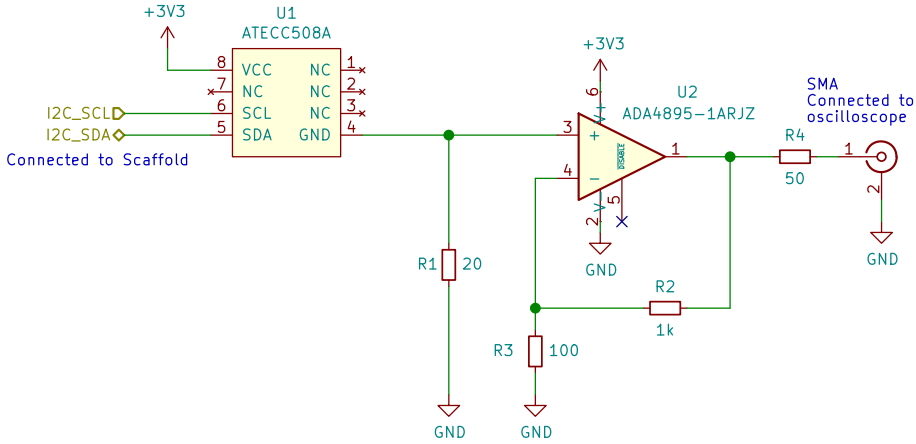

Fig. 8. Setup schematics for power trace measurement

Name	Value	Comments
Raw	`0x8f43`	Slot configuration value
Write config	encrypt	Writes are always encrypted
Write key	`0x3`	Write encryption key index
Read key	`0xf`	Read encryption key index
Is secret	True	This data slot can never be read
Encrypt read	False	Read are forbidden by "is secret" flag, but allowing plain text can help us if we manage to bypass "is secret" flag.
No MAC	False	MAC and HMAC commands with this data slot are allowed.

Table 2. Targeted data slot configuration details

attacks is extremely difficult: there is usually no way to know whenever a first-fault was successful or not until you manage to pass both of them! In this context, it's necessary to perform temporal and spatial scanning, injecting faults randomly and observing the different behaviors of the circuit.

In our case, we chose to attack the *Read Memory* command of the device. Our objective was to retrieve the first 32 bytes of the secret slot storing the wallet PIN hash, with the configuration described previously. We supposed the firmware would check for the "is secret" flag with a simple conditional branch to determine whether or not the user has rights to access this data slot. If this scenario is valid, only a single fault during the branch instruction should be enough to bypass the security.

Although the "encrypt read" flag shall not be relevant (all reads are forbidden!), it is a chance for us that it is set to False: if we manage to

bypass the first security check ("is secret" flag), we want the data to be output in plain and not encrypted with an unknown key.

Once the scenario is established, there are critical attack parameters that need to be found to perform successful security bypass of the *Read Memory* command: we need to know WHERE and WHEN to shoot with our laser. The next sections will explain how we searched for those settings.

5 Learning from the power trace

As we mentioned, the power trace of a circuit can reveal a lot of useful information. The Figure 9 shows a measured power trace of the circuit during the execution of the *Read Memory* command on an authorized data slot. The top blue waveform is the I^2C SDA signal, which transports the read command sent to the ATECC508A device. The bottom red waveform is the power trace. As soon as all the bytes of the command are received by the ATECC508A circuit, a rise in power consumption is visible: the CPU of the circuit starts processing the input command and thus requires more energy than when it was waiting. Once the command is processed, the power consumption goes back to an idle level.

When doing the same experiment but trying to access a forbidden data slot, the waveform in Figure 10 can be observed. We can see that the processing of the command takes less time: this is to be expected as the program returns an early error message when it checks for the access rights.

The observations can be improved by averaging the power traces to eliminate the noise from the clock jitter, and then superimposing both waveforms. The Figure 11 shows this measurement. The red waveform corresponds to the forbidden access, and the dark-gray waveform to the granted access. The beginnings of the power traces perfectly match until a precise time. This gives us a very precious information: it is the time when the circuit takes a different decision from the given inputs, when the program control flow differs. This time probably corresponds to a conditional branch testing the "is secret" flag of the data slot configuration.

The granted access power trace (dark-gray) also shows a repetitive pattern, shortly after the branch divergence. We supposed this part of the power trace corresponds to the data transfer from the EEPROM memory to a buffer in the RAM memory. We could validate this hypothesis experimentally by faulting the data transfer, injecting fault at this time when reading an authorized data slot. We observed that the faulted byte index was depending on which pattern we targeted, and we found out the

device copies the 32 bytes data slot by 8 transfers of 4 bytes. As expected this data transfer does not occur when the slot is secret (red power trace).

This power trace analysis gives us a precious hint on the fault injection time. It does not give an exact timing, but considerably reduces the search space, and therefore increases the probability of success.

6 Testing campaign

To search for vulnerabilities, we ran an automated one day long fault injection campaign on the *Read Memory* command. Each test was composed of the following steps:

1. Laser beam displacement: the laser is moved at a random location above the ROM memory region (see Figure 5). We didn't explore the whole chip with the laser and we chose to focus on the ROM memory as it usually gives good chances of faulting the program instructions during execution.
2. Laser pulse delay configuration: the fault injection time is randomized in a small time window around the branch identified in the power-trace (Figure 11).
3. ATECC circuit power-on
4. ATECC prelude commands for wake-up and initialization
5. Laser trigger activation: the next I^2C command sent to the circuit will send an electrical pulse to the pulse generator, which will activate the laser for a short duration and after a configured delay (set in step 2).
6. Execution of the *Read Memory* command. The laser illumination will occur during the processing of this command.
7. Log the response from the circuit (error code, returned data or communication errors)
8. Laser trigger deactivation
9. ATECC circuit power-off

Thanks to our custom I^2C communication management with our dedicated hardware, our script is able to raise an exception for every possible communication error. No fault can lead to attack script crashes: we are able to log any kind of chip misbehavior and let the test campaign run for a very long time.

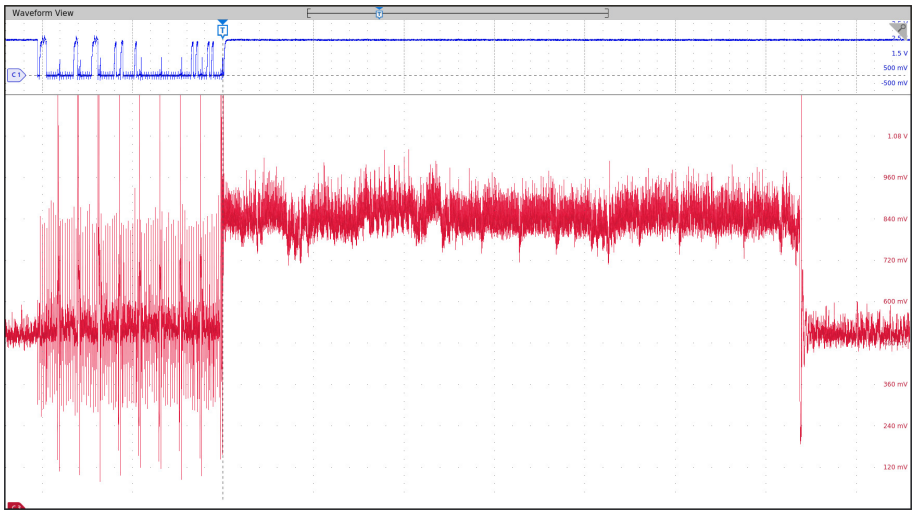

Fig. 9. Power trace during granted memory read

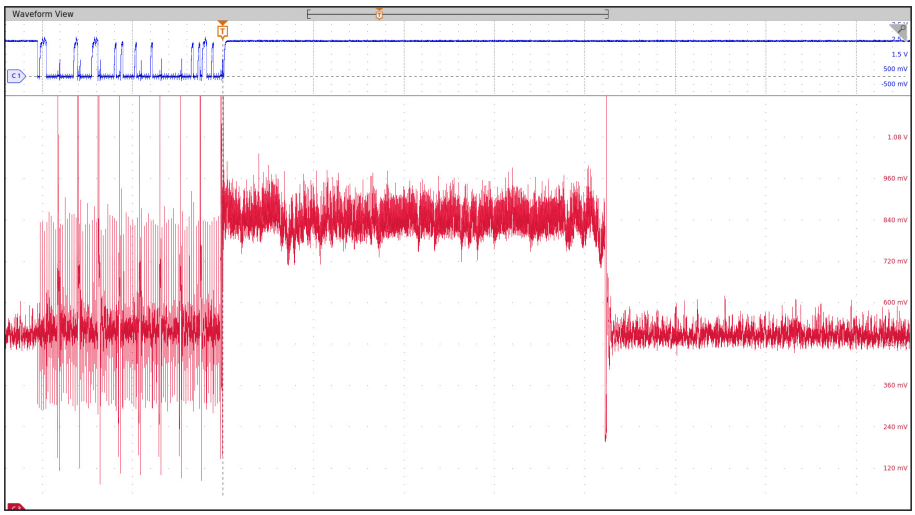

Fig. 10. Power trace during denied memory read

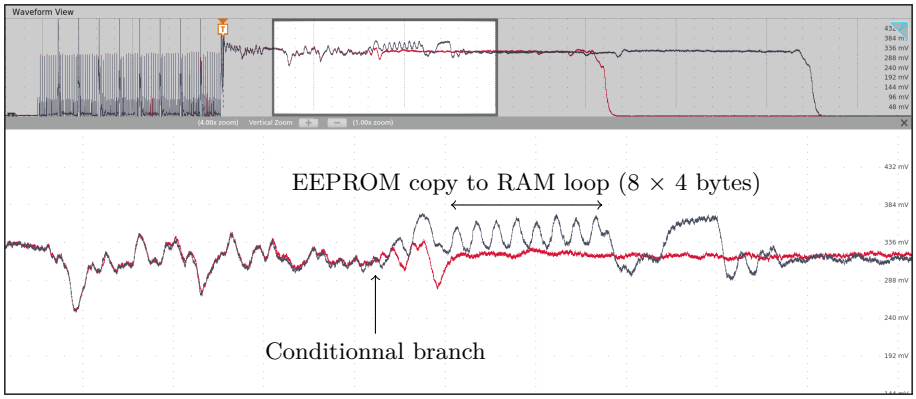

Fig. 11. Averaged power traces comparison between granted and denied read requests

7 Results

343617 fault injection tests have been performed during the campaign. 1546 tests resulted in data transmission from the ATECC508A device and the execution status "OK". Lots of different output data have been collected. The Table 3 is an extract of the most frequently received data. Most of the received data have the correct expected length (32 bytes), but unfortunately, none of them matched the initially programmed value `0123456789abcdef...`.

From the result log we observed some data was received multiple times. Although those records are random look-alike, since the device is restarted before every test, we deduced it was stored in the non-volatile memory of the device.

In Figure 12, we plotted the occurrences of the received data. The X-axis corresponds to the test number, growing over time. Some particular data are interesting and can be read as following:

- the data starting with `a712c613...` (line 1 in Table 3; line 1 in Figure 12) has been received 336 times between experiments n° 33 and n° 114317.
- the data starting with `a1ff80fa...` (line 4 in Table 3; line 5 in Figure 12) has been received 76 times between experiments n° 114613 and n° 147932.
- the data starting with `929b86e3...` (line 6 in Table 3; line 7 in Figure 12) has been received 58 times between experiments n° 148053 and n° 169969.

Count	Length	Output data from ATECC508A device
336	32	a712c6137b0b50b401d8deff8b0b3b8e5f2b01e0707d4eaeaeb6bbe589220274
152	32	a092cc6943e6c408bdd924e4ce90b8c895ddac03d2ada707088cace9d9cb803a
151	4	00000000
76	32	a1ff80fa7028066d4dcc023f23e2ec6b79864aa8b6e979e1d63cbf05277ebeb7
72	32	41e0f633a019cd625920691b11400c9387009e68d0b13e53d73257216a4c0ce8
58	32	929b86e3dff0ecb1d2318cf0c4bf5872b32d9db260cf012ae7c00d40cac19cc1
53	32	4e92d8096bfa78254581b9f5b987e60337e4f9860f92a2615581676e896854dd
51	32	011ffd4b459e81f8ab7f42cd2662fc6117cad15cb99155e72ed6b76211067e22
50	64	09c842000000000000000000000000000000000000000000000000000000000000000...
43	32	9dbf7427f5098feb2c708174875896f7294629a30049f5aa825dffa05b7c3c29
37	32	f6fecd81f528d1ebfcf005b0d59ebfd84839dbcc0c1a9614be3a13351009b107
31	32	8f8a22572231abafd8035be7d84eece928e7754d966b054fa4f02e5d02599bc6
29	32	069ff7317d731544177eb8d663f97f27dd3c7cbf1b41bc4e88eca06e41effc6c
21	32	c776a730a55dd031685d2afc76672ba5d23187ca07ce42b66286888be89cac2d
20	4	01000000
15	32	89f3c21a72ebb69fb1f6010fe3c0a3ab6ebb81356337b3e2a7024024d40ba371
14	32	2132c13ce836eda1ab62fc3c9b07345da28616d792e0ebc3e7bae5864c0d9e80
12	32	07f2bba24ebdd721e76b9e0d8e8b2b8431679a147f0562a8565cb382bf5ac2e1
12	32	e7edcd6b9e8c1c2ef387f529bc29cb7ccfe14ed4195d251a57525ba6f26870be
11	32	1c60381c2111566e7b200149b12bc72ee416bd90d1db927d4fe0abc008d0349a
11	32	487ce193a06c6fd01d38221f0fb1b5efaf3af73a8c3b1078732b34a03e10c806
9	32	3496bdbbe1653dfd789610c269d69f9dfbcc4d0ea9231a6367a001c752e5e097
9	32	e89fe351556fa969ef2625c714ee21ec7f05293ac67eb928d5e16be9114c288b
6	32	fea48df33529bd4490c47a7511d58cd367762ea3b99155e7d129489d11067e22
6	32	50f3f6d9cbbd00a75657998278f7783700d80b70c5f68d5d3e5a1fb2882dcd51
5	32	2987190f1ca47ae372a4c7d575272b066006a5f15f871e06249022da9a7de790
5	32	1856bdd3ee3b2092c83ccd918b9ebbcebf5db12b195d251aa8ada459f26870be
4	32	f849cb1a3e0aeb9ddcd0a6b5b93c5b3641db65eb7f0562a8a9a34c7dbf5ac2e1
4	32	2ffef9424c7e67d31b519d3d4ea96444265a5189aadba8ab27624ca34c2fdf27
4	32	ee71dacb9ef9be9cb79fbe2da2d87200e7db278645d70b31f34b3827a634b450
4	32	617b9c689bc8017dc2fab6cedb0eec4a9ad05776ce2259ffe2e6840c70b8d07b
4	32	9ec0ecd0eb7f3dc1f9418e76ecb99cf8ea6ca889ce2259ff1d197bf370b8d07b
3	32	ba6d03441f848722d1879d7b1c4200c1acf76d2c18378934cf80a74790f448cc
2	32	39f89fdcf322080d983818ce187b8080a8368ce9c3358cba4dc2fbe281ecd863
2	32	873ffa64d6a33ced01c5600e366ef3b2ea67c66b5b0cf08a67dae901d429e2d6
2	32	172493e925d895d5d49d1d7f2359515e0fb9d6c5c67eb9282a1e9416114c288b
2	32	f92487890dc429f82cc5806e544e0f95ad8083401b41bc4e77135f9141effc6c
1	32	03ef60a44cc20e776047a0fa7824021d32a8c80804cd330b1a5177d9b58f3264
1	32	e94af4f13db169fd5ff6f20fb347eccb424ed1421dd2bdb1ecf63bae66f67d5c
1	25	09c84200000000000000000000000000000000000000000000

Table 3. Collected output data from ATECC508A device resulting from *Read Memory* command fault injection

— etc.

For the mentioned data, the occurrence ranges intersection is $\emptyset$. From this, we understood the laser faults injected during the tests were changing the data stored in the non-volatile EEPROM memory. It is probable that a laser fault injected between experiments n^o 114317 and n^o 147932 overwritten `a712c613` with `a1ff80fa`. This pattern can be observed 12 times across all the testing campaign.

Furthermore, some data seem to come in pairs:

— `a1ff80fa...` with `07f2bba2...` (Pair A in Figure 12)
— `929b86e3...` with `3496bdbb...` (Pair B in Figure 12)
— `f6fecd81...` with `50f3f6d9...` (Pair C in Figure 12)
— `41e0f633...` with `e7edcd6b...` (Pair D in Figure 12)
— etc.

We supposed pairs were corresponding to the same data, which were returned encrypted in some cases, and in plain text in the other cases. We could not verify this hypothesis since the encryption key was unknown to us.

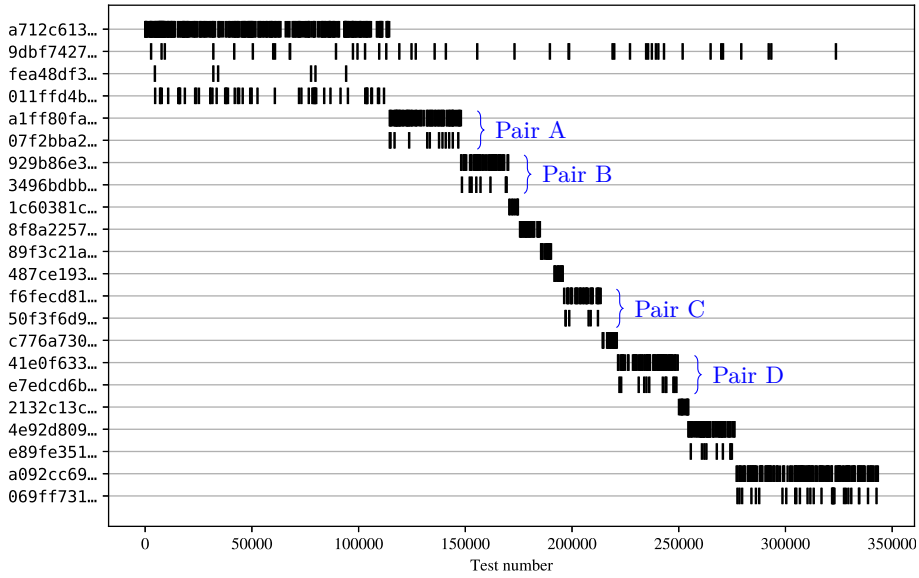

Fig. 12. Output data occurrences over the experiments

At the end of the campaign and after the results analysis, we believed the data we were trying to read in the secret slot had been overwritten. The

ATECC508A circuit provides the *"GenKey"* command which generates a new P256 elliptic curve key (32 bytes) and stores it in the data slot given in the command parameters. A laser fault injection had probably disturbed the program control flow and executed this command by accident.

The ATECC508A device also provides a *"MAC"* command which computes and returns the SHA-256 digest of random nonces, the device serial number, and the first 32 bytes of a chosen data slot. We used this command to demonstrate that the last received data in the test log (`069ff731...`) was the content of the targeted data slot, proving we had several successful faults during the whole campaign.

To summarize, the possible circuit behavior after fault injection during our testing campaign were the following:

- The *"Read"* command is executed but the arguments are faulted and an authorized data slot is returned instead of the requested one.
- The command executed is not the correct one: this can happen if the command code is faulted, or if the command dispatcher control flow is modified. For instance, if the *Random* command is executed instead of the *Read* command, the circuit will return an "OK" status with generated random data.
- The command executed is not the correct one and overwrites the data we want to read. This happens if a key generation is accidentally started. This is a very problematic situation as it is hard to detect and usually requires changing the slot to be read or replacing the sample by a new one.
- The fault triggers an invalid write in the EEPROM configuration memory. This may destroy the chip and require replacing it. We encountered this case a few times, and this is the most annoying result since we need to prepare a new sample to continue the tests.
- Command execution is faulted and produces an internal checking error.
- And other behaviors very hard to understand in a black-box approach!
- The *"Read"* command is executed but the data sent seems to be incorrect, probably overwritten.

8 Refining the attack

The attack campaign detailed previously showed that there might be a risk to erase the targeted asset data before being able to retrieve it. We

spent more time to refine the attack parameters to reduce the risk of data loss. We identified a precise timing and laser beam position for which the chances of success of the attack is high, and the chance of data lost is low.

We tested those new parameters on a new device and we were able to extract the expected data (`0123456789abcd...`) in less than 2 minutes of testing. This demonstrated that the vulnerability can be exploited in a real-case scenario.

9 Further work

During our security evaluation of the ATECC508A circuit, we also identified another vulnerability which allowed us to change the serial number of a circuit. We also demonstrated this vulnerability can be used to unlock the configuration zone of a locked circuit, which may grant access to all the stored data. Those attacks were really hard to perform compared to the one presented in this article, and we only managed to perform it twice. Furthermore, there is a very high risk of destroying the chip permanently when attempting to unlock it, making the attack not very practical for the moment. We are still investigating on those attack paths.

Microchip released the ATECC608A, which is the backward-compatible successor of the ATECC508A. This circuit seems to use the exact same silicon, with a new firmware providing more functionalities, and with more software security hardening. A security evaluation of this circuit against fault injection should be interesting to perform, and for sure a real challenge!

10 Conclusion

We identified the *Read* vulnerability in less than one month of work and demonstrated it is a practical attack. Although we used expensive equipment, the gain from such an attack can be very high, in particular if the target is a stolen hardware wallet.

Sample preparation is limited since it is not mandatory to thin the silicon substrate. Since the samples can be easily acquired and do not require a lot of preparation, we did not hesitate to inject faults with a lot of power or pulse count in order to increase the success probability. Therefore, two chips have been destroyed during the laser campaigns before getting a successful breaking fault. Destruction may come from

invalid EEPROM write operations at critical addresses, but we could not verify this hypothesis.

A particular difficulty we met when researching exploitable faults on the ATECC508A is that this circuit cannot be programmed back to default factory settings. It is not either possible to load custom code for testing purpose. Some chips were broken after bad EEPROM configuration, due to misunderstanding of the datasheet while discovering the circuit and its commands. Other chips have been broken with invalid writes in EEPROM memory, induced by laser faults. During the *Read* attack campaign, known data which have been loaded during configuration and before locking the chip have sometimes been overwritten by undesired key generation induced by faults - making the detection of a successful fault undetectable since the dumped data was not the expected one. Every time a chip is broken, a lot of time is spent preparing another sample. This slows down attackers in finding vulnerabilities.

Vulnerabilities on standard microcontrollers usually give full access to all stored data. Regarding the ATECC508A, our vulnerability is only applicable to a specific data slot configuration. Data slots configured for P256 key storage, which is the typical use case for this product in IoT, are not vulnerable to this attack path.

As of today we consider this chip vulnerable to laser fault injection. Despite the identified vulnerabilities, we think the ATECC508A circuit was a smart choice to protect secrets and we want to remind it is a much safer security solution than relying on microcontrollers non-volatile memory for storing secrets. No agreement with the manufacturer is required to buy and develop a product using this circuit. However, the security assurance level of this solution is not as high as certified secure elements.

11 Acknowledgments

The vulnerabilities presented in this paper have been reported to Microchip - the ATECC508A manufacturer - before any publication. We want to thank them for their warm collaboration and their will to fix the issues and make their products more secure. This work have also been reported to Coinkite - the Coldcard wallet manufacturer.

References

1. Karim Abdellatif, Charles Guillemet, and Olivier Hériveaux. Unfixable Seed Extraction on Trezor - A practical and reliable attack. `https://donjon.ledger.com/Unfixable-Key-Extraction-Attack-on-Trezor/`, 2019.

2. Ledger Donjon. Scaffold. `https://github.com/Ledger-Donjon/scaffold`, 2019.
3. Daniel Genkin, Adi Shamir, and Eran Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. *Advances in Cryptology - CRYPTO' 2014*, pages 444–461, 2014.
4. JIL. Common Criteria Part 3: Security assurance components. `https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf`, 2012.
5. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. *Advances in Cryptology - CRYPTO' 99*, pages 388–397, 1999.
6. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113, 1996.
7. Kraken. Inside Kraken Security Labs: Flaw Found in Keepkey Crypto Hardware Wallet. `https://blog.kraken.com/post/3248/flaw-found-in-keepkey-crypto-hardware-wallet-part-2/`, 2020.
8. Microchip. ATECC508A CryptoAuthentication Device Complete Datasheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/20005927A.pdf`, 2019.
9. Johannes Obermaier and Stefan Tatschner. Shedding too much Light on a Microcontroller's Firmware Protection. *11th USENIX Workshop on Offensive Technologies - WOOT 17*, 2017.
10. Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. *Smart Card Programming and Security*, pages 200–210, 2001.
11. Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple Photonic Emission Analysis of AES. *Cryptographic Hardware and Embedded Systems - CHES 2012*, pages 41–57, 2012.
12. Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. Optical Fault Attacks on AES: A Threat in Violet. *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC*, pages 13–22, 2009.
13. Sergei P. Skorobogatov. Semi-invasive attacks - A new approach to hardware security analysis. 2005.
14. Sergei P. Skorobogatov. Optical Fault Masking Attacks. *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 23–29, 2010.
15. Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. pages 2–12. Springer, 2003.
16. Christopher Tarnovsky. Security Failures In Secure Devices. *Black Hat DC*, 2008.
17. Jasper. G. J. van Woudenberg, Marc F. Witteman, and Frederico Menarini. Practical optical fault injection on secure microcontrollers. *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 91–99, 2011.