How Primitive but How Effective: Fault-Injection Attack on Cryptographic Accelerator of Microchip CEC 1702 Microcontroller

Lukáš Daněk, Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
{daneklu2|novotnym}@fit.cvut.cz

Abstract—**Fault injection attacks pose a substantial threat to the security of digital systems, compromising integrity and exposing vulnerabilities. This article explores fault injection techniques, specifically voltage glitching, within the context of Lenstra's attack, which is an extension of the Bellcore attack, on RSA-CRT, which has existed for decades. The focus is on the cryptographic accelerator of the Microchip CEC 1702 microcontroller. The study employs the ChipWhisperer toolkit to perform fault injection attacks on both software and hardware implementations of RSA-CRT. Results reveal vulnerabilities in the commercially produced Microchip CEC 1702 microcontroller, highlighting potential security risks associated with fault injection attacks.**

Index Terms—**Fault-injection analysis, Bellcore attack, Lenstra's attack, CEC 1702, ChipWhisperer**

I. Introduction

In the field of cybersecurity, fault injection attacks pose a significant threat to the integrity of digital systems. This paper discusses fault injection techniques, with a focus on voltage glitching, which targets the vulnerability of the cryptographic accelerator in the CEC 1702 MCU [9]. The CEC 1702 is an ARM Cortex-M4-based microcontroller manufactured by Microchip that provides a number of hardware-supported cryptographic functions, such as hardware accelerators for AES, RSA and ECC, a cryptographic hash engine, a random number generator, and more. These cryptographic functions must meet strict requirements for resistance to attacks. To the best of our knowledge, only the resilience of the AES accelerator against Correlation Power Analysis (CPA) was evaluated in [8]; the authors found no vulnerabilities in this case. In this paper, we focus on the RSA hardware accelerator and mount a fault-injection attack.

Fault-injection attacks were known already in the 1990's. In 1997, the group of researchers from Bellcore showed how one might exploit the deliberately introduced faults to reveal secrets if utilizing the Chinese Remainder Theorem (CRT) to generate RSA signature [5]; Lenstra later improved their attack. In the same year, Biham and Shamir introduced the Differential Fault Analysis. They demonstrated their method on DES [3]; it was adopted to AES by Dusart et al. [7]. Fault injection attacks include also the Collision Fault Analysis [4] or Fault Sensitivity Analysis [10]. They can also be combined with passive attacks [6], [21].

This research has been supported by the grant VJ02010010 of the Ministry of the Interior of the Czech Republic, "Tools for AI-enhanced Security Verification of Cryptographic Devices" in the program Impakt1 (2022-2025).

This paper showcases Lenstra's attack, which is an extension of the Bellcore attack [5] on RSA-CRT [20], as a practical example that demonstrates how deliberately injected faults can compromise even the security of commercially produced Microchip CEC 1702 microcontroller. The attacks are carried out using the ChipWhisperer [12] tools, which are designed to simplify side-channel attacks.

This paper is structured as follows: In Section II, we discuss fault injection attacks and we summarize the Bellcore attack on RSA implementation using the CRT that was later improved by Lenstra. In Section III, we describe the measurement setup, and in the next Section IV, we bring the results of our measurements. In the last Section V, we conclude the paper.

II. Fault Injection attacks

Fault Injection attacks involve the intentional manipulation of system behavior by introducing controlled faults into system operations. Such vulnerabilities can be exploited to compromise security mechanisms, gain unauthorized access, or extract sensitive information. This section delves into one fault injection technique – voltage glitching – and Lenstra's attack on RSA-CRT, which exploits injected faults to obtain the private keys of the RSA [20] asymmetric cipher.

A. Voltage glitching

Voltage glitching [2], [22] is a fault injection method where the chip's supply voltage is intentionally reduced for a brief period of time. This deliberate voltage drop results in decreased current flow and subsequently slows down the charging of parasitic capacities, leading to delayed data propagation. This effect is akin to clock glitching, where data is not fully prepared by the time the clock reaches its active edge. Unlike clock glitching, voltage glitching doesn't necessitate a device to be externally clocked, eliminating a potential disadvantage.

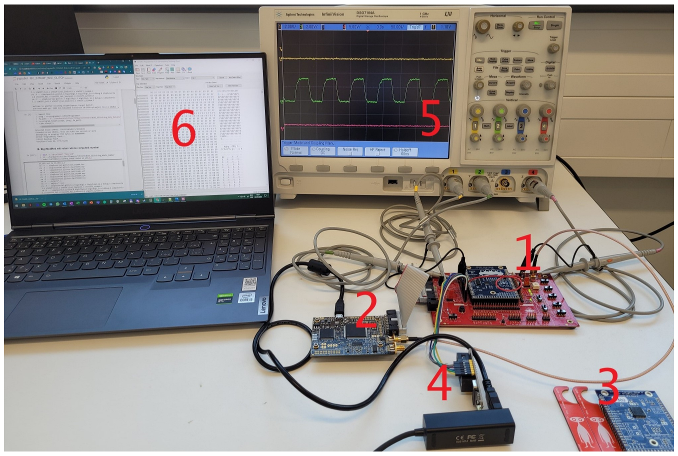

Fig. 1. Setup for fault injection on CEC 1702 using ChipWhisperer Lite. 1—CEC 1702 inserted into the UFO board with connected oscilloscope probes, 2—ChipWhisperer Lite, 3—STM32F3 spare Target, 4—USB-SPI CH341 programmer connected to UFO board and a PC via USB, 5—Oscilloscope Agilent MSO 7104A, 6—PC running the ChipWhisperer environment.

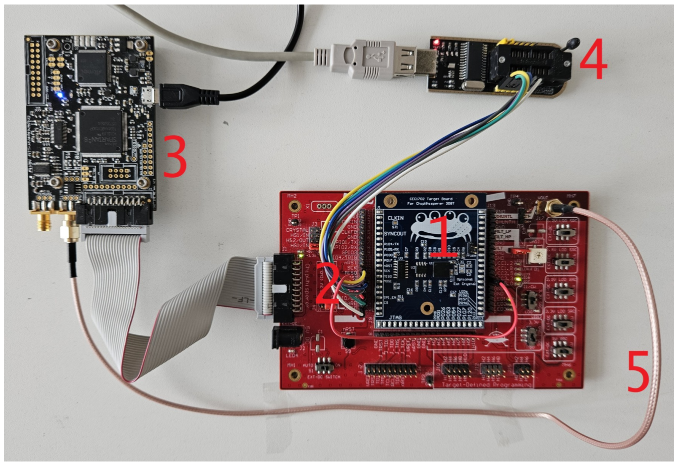

Fig. 2. Detail of CEC 1702 connection to ChipWhisperer-Lite and USB-SPI convertor. 1—CEC 1702 inserted to UFO board, 2—UFO Board, 3—ChipWhisperer Lite connected to UFO Board using 20-wire cable, 4—USB-SPI CH341 connected to UFO board, 5—50 cm coaxial cable interconnecting UFO Board power circuit with ChipWhisperer-Lite glitch generator.

However, voltage glitching can be more challenging to fine-tune, and there's a risk of permanent damage to the device if not executed precisely.

B. *Lenstra's attack on RSA-CRT signing or decryption*

The Lenstra's attack is an extension of a Bellcore attack [5]. It utilizes the fact that RSA signing/decryption using CRT replaces one exponentiation modulo n with two exponentiations modulo p and q, where $n = p \cdot q$ is a public modulus and p and q are secret prime numbers. Using CRT accelerates RSA signing/decryption by approximately four times.

Introducing an error into the computation when working with, e.g., prime number p, allows to calculate the second prime number q as $q = \gcd((c - m')^e, n)$, where e is a public exponent, c is an *encrypted message* or a *message to be signed* (input to the algorithm) and m' is a *decrypted message* or a *signature* with a fault (faulty output of the algorithm). Revealing the value of q will break the system, as we get p as $p = n/q$ and the secret private key d would be then computed from the public key e by the formula $d = e^{-1} \mod ((p-1) \cdot (q-1))$.

III. Measurement Setup

All attacks were conducted using the ChipWhisperer suite [14]. The attacks were carried out on the CEC 1702 MCU [18] and two versions of RSA-CRT, one implemented purely in software (SW) and the other utilizing the MCU's hardware accelerator (HW). Fine-tuning of the power glitch generator parameters was necessary, and for this purpose, the employed program performed only modular reduction, a crucial component used in RSA-CRT.

A. *Hardware and tools used*

The toolkit used for this research was the comprehensive open-source ChipWhisperer suite, designed for side-channel attack profiling on embedded devices and for assessing their resilience against such attacks. Within the scope of this study, the toolkit was employed to conduct power glitching attacks.

Firstly, we utilized the STM32F3 [19] target MCU. Tutorials for performing fault injection attacks on the STM32F3 were already available, making it a convenient choice during the preparatory phase. Subsequently, we selected an additional target, the CEC 1702 [9], ARM Cortex-M4 based, equipped with a hardware cryptographic accelerator. Importantly, no prior fault injection attacks had been conducted on the CEC 1702 using the ChipWhisperer toolkit. Programming the CEC 1702 directly into its flash memory was achieved through a USB-SPI converter [11]. Both targets were controlled via the ChipWhisperer API, which also managed the synchronization of operations between the MCUs and the glitch generator.

To mount the attack, we used the following items:

1) ChipWhisperer-Lite Capture Board [16] serving as a glitch generator,
2) UFO board for module integration [17],
3) NewAE CEC 1702 target module [18] for UFO Board (referred to as CEC 1702 hereafter),
4) NewAE STM32F3 target module [19] for UFO Board (referred to as STM32F3 hereafter),
5) 50 cm coaxial cable with SMA-F connectors from the ChipWhisperer *SCAPACK-L1* kit [14] for introducing glitches into the power supply,
6) USB-SPI converter *CH341A* [11] for programming the Flash memory of the CEC 1702 module,
7) Agilent MSO 7104A oscilloscope [1] for monitoring signal waveforms,
8) Lenovo Legion S7 personal notebook.

All the hardware used is depicted in Figure 1, and detailed connections for the CEC 1702 and its connection through the UFO board is shown in Figure 2.

```
1  // Prepare BUFF8 structures accepted by functions
       performing the PKE operations
2  BUFF8 p_buff8_array, c_buff8_array,...;
3  ...
4  // Load RSA key int HW accelator memory
5  rsa_load_key(RSA_BITS, &d_buff8_array, &
       n_buff8_array, &e_buff8_array, BIG_ENDIAN);
6  ...
7  while(pke_busy() == 1) asm nop;
8  // Load remaining values needed to perform RSA-CRT
9  rsa_status = rsa_load_crt_params(RSA_BITS, &
       dp_buff8_array, &dq_buff8_array, &qi_buff8_array
       , BIG_ENDIAN);
10 ...
11 while(pke_busy() == 1) asm nop;
12 // Signal to ChipWhisperer to start glitch generator
13 trigger_high();
14 // Perform decryption
15 rsa_status = rsa_crt_decrypt(RSA_BITS, &
       c_buff8_array, BIG_ENDIAN);
16 pke_start(0);
17 ...
18 while(pke_busy() == 1) asm nop;
19 // Signal to ChipWhisperer that decryption finished
20 trigger_low();
21 // Read decprypted messsage
22 bytes_read = pke_read_scm(&plaintext_byte_array[0],
       RSA_BYTES, 5, BIG_ENDIAN);
```

Fig. 3. Simplified implementation of RSA-CRT decryption using HW accelerator API calls. Decryption is performed on lines 15–18.

A custom implementation of RSA-CRT was created to facilitate the transformation of the attack findings from STM32F3 to CEC 1702. This involved creating two versions using a 1024-bit hardcoded key: one relied solely on a pure software implementation without any security safeguards, while the other leveraged the CEC 1702 hardware accelerator exclusively for cryptography operations. The compilation of the executables was achieved using the MikroC IDE with default compiler settings, which provided an integrated API for invoking hardware accelerator operations. Simplified implementation utilizing HW accelerator API calls is provided in Figure 3.

1) Voltage glitch generation: As illustrated in Figure 4, glitches are created using the MOSFET transistor, which is part of the ChipWhisperer-Lite. The transistor is adequately oversized to prevent its destruction. To introduce glitches into the power supply, a coaxial cable with SMA-F connectors is used; the interconnection is displayed in Figure 2. Opting for a different cable length could necessitate searching for an alternative glitch generator configuration in the future.

All glitch generator parameters are described in Figure 5. The glitch generator parameters' ranges are as follows:

- The `WIDTH` parameter within the range of -49.8 to 49.8.
- The `OFFSET` parameter within the range of -45 to 45.
- The `REPEAT` parameter within the range of 1 to 2^{32}.
- The `SHIFT` parameter within the range of 0 to 2^{32}.

Values of `OFFSET` and `WIDTH` can be rational numbers, but only integer numbers are considered in this article. The introduced glitch is derived from the internal CLK source in ChipWhisperer-Lite or an external source (see CLK_{sync} in Figure 4). The timing of glitch injection is controlled during attacks by the `TRIG` signal. For ease of attack, `TRIG` is set by the program to an active state when the computation phase that needs to be influenced is being executed. Glitches should be injected consecutively for a given number of clock cycles (`REPEAT` parameter) instead of introducing one glitch lasting several clock cycles to achieve sufficient power supply fluctuations while avoiding possible destruction of the device.

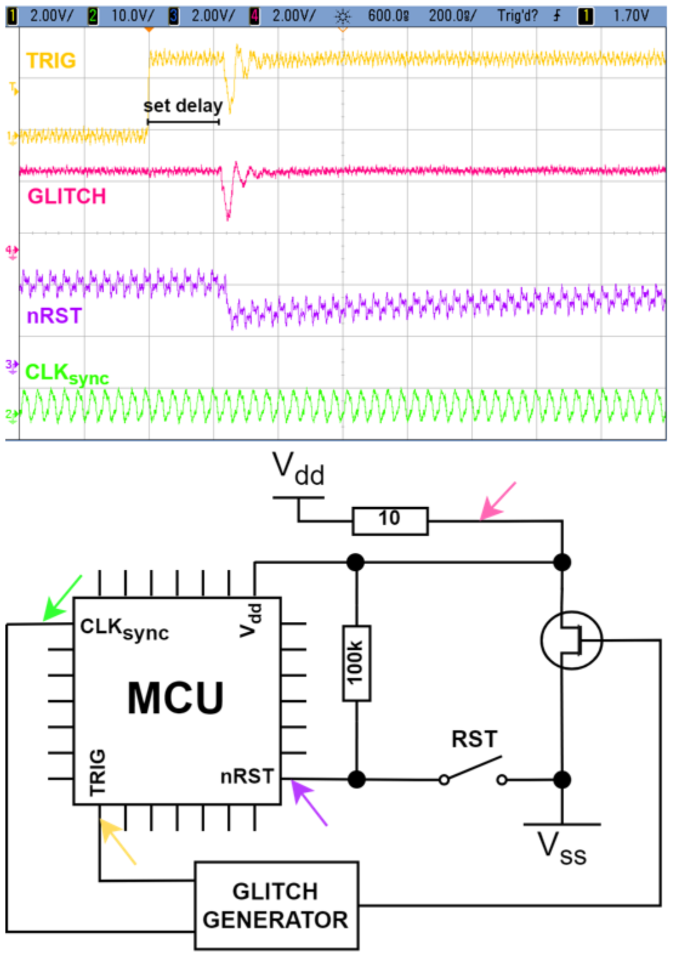

Fig. 4. Simplified glitch scheme with power traces demonstrating voltage glitch injection.

The configuration for the CEC 1702 and ChipWhisperer-Lite is as follows:

1) CEC 1702 runs at its default setting of 48 MHz.
2) CLK_{sync}, as shown in Figure 4, is set to 12 MHz using internal MCU's PWM.
3) ChipWhisperer-Lite multiplies the input CLK_{sync} by 4 to create a 48 MHz clock, allowing for glitch injection timing granularity roughly matching the MCU frequency.
4) ChipWhisperer-Lite communicates with the MCU via the SimpleSerial protocol [13], utilizing UART at 38 400 Baud.

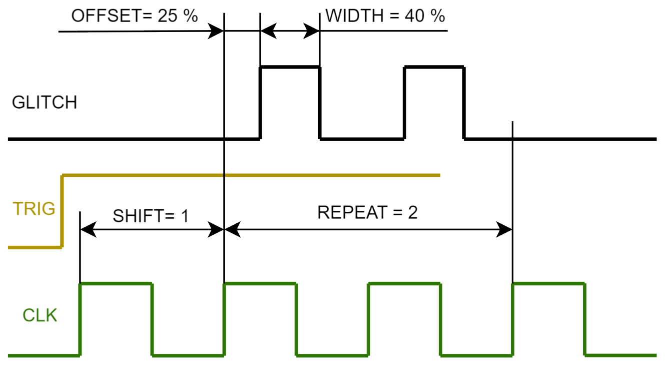

Fig. 5. Overview of glitch generator parameters. `OFFSET` and `WIDTH` parameters units are percentages of the generator's CLK period width. `OFFSET` can have negative values, then the offset is calculated as `100 - OFFSET`. Glitches can be injected consecutively for a given number of clock cycles set by the parameter `REPEAT`. The parameter tells how many times a glitch should be introduced in a row. `SHIFT` parameter corresponds to `ext_offset` parameter in ChipWhisperer API and tells how many CLK cycles should be glitch introduction postponed (shifted).

B. SW implementation of RSA-CRT and voltage glitching

Protection against side-channel attacks can be carried out at different levels. One can protect, e.g., the whole processor or just the cryptographic accelerator. To distinguish whether and at what level the protections are implemented, first, only a software implementation of a modular reduction used in RSA-CRT is evaluated. This simple operation takes much less time than the entire software implementation of RSA-CRT, and therefore it is possible to easily test many combinations of glitch generator parameters, which are depicted in Figure 5. As CEC 1702 is clocked by an internal oscillator and does not allow an external clock signal, we performed only a voltage glitching attack. The glitch parameters were described in Section III-A1. It is crucial to set the correct number of glitch repetitions, as they must be repeated consecutively to induce a sufficient power drop. This is especially important because, at such a high synchronization frequency, glitches with a maximum duration of 10.375 ns are negligible.

Glitch injection is tested at various execution times by specifying a `SHIFT` in clock cycles from the activation of the trigger `TRIG`, as illustrated in Figure 4.

If a glitch is introduced into the power supply in such a way that it affects not only the values in the regular registers but also the program counter or peripheral configuration registers, the device may enter an undefined state, cease communication with ChipWhisperer, and require a restart. This restart takes approximately 1 second, significantly slowing down the process of finding the correct glitch generator configuration for successful glitching. Therefore, it is essential to minimize the number of device restarts and, if necessary, reduce the intensity of glitch injections to avoid such disruptions.

1) Attack on the single operation: To tune the system, i.e., to find the best combination of glitching parameters (see Figure 4), we initially focused on introducing the glitches when the SW implementation of RSA-CRT is calculating

TABLE I
OPERATIONS TIMES – OVERVIEW OF OPERATION TIMES WITH DIFFERENT IMPLEMENTATIONS.

Operation		Bit length	Frequency	Duration
$c_p = c \mod p$	SW	256 (128)	48 MHz	67.27 μs
$c_p = c \mod p$	SW	1024 (512)	48 MHz	916.13 μs
RSA-CRT	SW	1024	48 MHz	6.46 s
RSA-CRT	HW	1024	96 MHz	3.23 ms

the remainder $c_p = c \mod p$ (c is the ciphertext or the message to be signed). First, the standalone function responsible for reducing a 256-bit number by a 128-bit prime number was tested to determine whether introducing glitches could modify the result. Then, we did the same for a 1024-bit number reduced by a 512-bit prime.

Focusing only on a part of the calculation and using a shorter bit length allows for testing a larger range of glitch generator configurations. This is because a single computation takes significantly less time than the entire decryption process in software, as indicated in Table I. To reduce the time spent on data transfer between the MCU and ChipWhisperer via UART, the MCU receives only a 1-byte command to initiate the 128-bit reduction and returns only a 1-byte response indicating whether the computation was modified (1) or not (0).

2) Attack on the whole encryption: The RSA-CRT implementation accepts only a 1-byte command to perform decryption using hardcoded data with a hardcoded key. It then returns the entire 1024-bit message, which is necessary to calculate the secret key stored in the MCU if the encryption is successfully modified.

Since encryption operation takes much longer than doing the modular reduction itself (see Table I), glitch injection is performed with generator settings that have proven to be the most effective in terms of the ratio of successful glitch injections to device restarts due to errors.

Glitches are introduced when the 1024-bit to 512-bit modular reduction is performed. However, based on the evaluation of the attack on the 256-bit operation alone, glitches are only introduced at certain interval of the operation time to save time during the attack.

C. RSA-CRT using hardware cryptographic accelerator and voltage glitching

The code in CEC 1702 was modified accordingly: The software implementation of RSA-CRT was replaced with a function call of the RSA hardware accelerator. Simplified implementation is in Figure 3, and the attack is focused only on operation `rsa_crt_decrypt()`. The glitch generator settings are the same as those for SW implementation, as it was tested that they modify the computation process in MCU. The 48 MHz frequency of MCU is kept for synchronization even though the HW accelerator internally operates with double 96 MHz frequency.

To verify that there is no correlation between the glitch generator settings and the values of the RSA key, a glitching attack

TABLE II
OVERVIEW OF PERFORMED GLITCHING ATTACKS. ONLY THE RESULTS OF THE MOST SUCCESSFUL GLITCH GENERATOR SETTING ARE PRESENTED FOR EACH ATTACK. FOR AN EXPLANATION OF PARAMETERS (W) IDTH, (O) FFSET, (R) EPEAT, AND SHIFT, PLEASE SEE FIGURE 5.

Operation	Length (bits)	Impl.	Key	MCU	Glitch settings			Total SHIFT count	Results (%)		Ratio r_s
					W	*O*	*R*		*Success*	*Reset*	
$c_p = c \mod p$	256	SW	—	MCU1	48	-44	9	3 930	0.38	0.13	3
RSA-CRT 1024bit	1 024	SW	key1	MCU1	46	-41	11	11 212	0.50	0.86	0.58
RSA-CRT 1024bit	1 024	HW	key1	MCU1	46	-41	11	75 000	45.69	7.52	6.07
RSA-CRT 1024bit	1 024	HW	key2	MCU1	46	-41	11	75 000	39.63	4.37	9.08
RSA-CRT 1024bit	1 024	HW	key3	MCU1	46	-41	11	75 000	40.04	6.13	6.53
RSA-CRT 1024bit	1 024	HW	key1	MCU1	46	-41	11	75 000	45.69	7.52	6.07
RSA-CRT 1024bit	1 024	HW	key1	MCU2	46	-41	11	75 000	41.47	2.79	14.87
RSA-CRT 1024bit	1 024	HW	key1	MCU3	46	-41	11	75 000	49.84	3.38	14.73

is repeated for three different encryption keys and the same glitch generator settings, and the success rate of the attack is then evaluated.

Glitching was carried out with the same generator settings on three different CEC 1702 targets to test whether the glitch response is consistent among the whole CEC 1702 series.

IV. FAULT INJECTION RESULTS

The evaluation was performed on a notebook PC equipped with an Intel i5-10300H CPU, RAM with a capacity of 16 GB, and SSD with a capacity of 512 GB. ChipWhisperer environment version 5.5.0 [15] is used. ChipWhisperer-Lite capture board runs on firmware version 0.30.0.

An overview of the results of all performed attacks is presented in Table II. One full glitching attack consists of repeating the glitching on a range of different shifts with a fixed WIDTH, OFFSET, and REPEAT parameters, denoted as W, O and R in the Table II. We used the combination of glitching parameters that proved to provide the best results (see Section IV-A1 and Table III). Modular reduction attack was performed on a SHIFT range 0 – 3 929, RSA-CRT in SW on a SHIFT range 11 166 – 22 377, and RSA-CRT on a SHIFT range 0 – 74 999, where each SHIFT value in each range was used once.

Results represent the percentage of successful glitches. We also present the percentage of device resets, as they are undesired events prolonging the whole attack. Ratio r_s represents a ratio between successful glitches and the number of resets of the device for one full glitch attack.

A. SW implementation of RSA-CRT and voltage glitching

1) Attack on the single operation: Performing the attack on a single operation, which is modular reduction, narrows down the possible values of WIDTH, OFFSET and REPEAT suitable for further attacks on RSA-CRT. Modular reduction is less time consuming and therefore more combinations of these parameters can be tested.

One modular reduction of a 256-bit long number takes 3929 ticks. With one glitch generator configuration, it is necessary to attempt glitch injections in every individual clock cycle (SHIFT). The pulse width (value of WIDTH parameter) should be as large as possible. Only negative OFFSET relative to the rising edge of clock CLK is tested; a glitch, therefore, starts before the rising edge occurs (MCUs register operations begin) and lasts beyond the rising edge. For each pulse width, there is a maximum limit on the number of pulse repetitions (REPEAT parameter value) beyond which the device enters undefined states and does not complete the calculation. Two areas of generator parameter testing were conducted, with a granularity of one step (each parameter is incremented by 1):

1) WIDTH = 45, 46, 47, 48; OFFSET = -44, ..., -40; REPEAT = 10, 11, 12
2) WIDTH = 47, 48, 49; OFFSET = -44, ..., -40; REPEAT = 7, 8, 9

A total of 105 parameter combinations were tested, and out of these, 32 combinations resulted in successful glitch injections, leading to calculation modifications. Parameter combinations with the highest success rates $r_s = \frac{\#successful\ glitches}{\#resets\ of\ device}$ are listed in Table III. The ratio r_s of successful glitches to the number of device restarts is used as a main criterion for comparing results, rather than a ratio of successful glitches and the number of performed glitches. Many device resets indicate that glitches are too strong and significantly prolong the glitch parameters test due to the overhead of waiting for timeout and device restart.

2) Attack on the whole encryption: One RSA-CRT signing using a 1024bit key takes approx. 6 seconds and modular reduction to a 512bit number in the algorithm takes 916.13 μs (43 974 ticks), see Table I (after omitting the overhead with sending and checking messages in Python). Testing all the sets of glitch parameters as in Subsection IV-A1 in the entire range of possible shifts would be very time-consuming. Therefore, only the most successful settings of the glitch generator were used from each set, highlighted in Table III.

The range of shifts, where glitches are introduced, was interpolated from the interval of SHIFT parameter with the most successful glitches of 256bit modular reduction represented in Fig. 6. The SHIFT range 1 000 – 2 000 (25.4 % and 50.9 % of maximum value 3 930) was interpolated to the range 11 166 – 22 377 (25.4 % and 50.9 % of maximum value 43 974).

Although only two generator configurations were tested and the range was limited, the testing of one generator setting with 11 212 different shifts took approximately 18:48 h. The results of the glitch attack are presented in Table IV. Both

TABLE III

SINGLE MODULAR REDUCTION ATTACK RESULTS FROM TESTING TWO GROUPS OF GLITCH GENERATOR PARAMETERS DESCRIBED IN SECTION IV-A1. ONLY THE MOST SUCCESSFUL COMBINATIONS OF PARAMETERS FROM EACH GROUP ARE LISTED. FOR AN EXPLANATION OF PARAMETERS `WIDTH`, `OFFSET`, AND `REPEAT`, PLEASE SEE FIGURE 5.

WIDTH	**OFFSET**	**REPEAT**	**Success**	**Reset**	**Normal**	**Ratio r_s**
46	**-41**	**11**	10	43	3 877	0.023
46	-43	12	30	1 528	2 372	0.002
47	-41	10	21	96	3 813	0.022
47	-42	10	28	138	3 764	0.020
48	**-44**	**9**	15	5	3 910	3
48	-40	9	14	15	3 901	0.093
48	-43	9	12	15	3 903	0.08

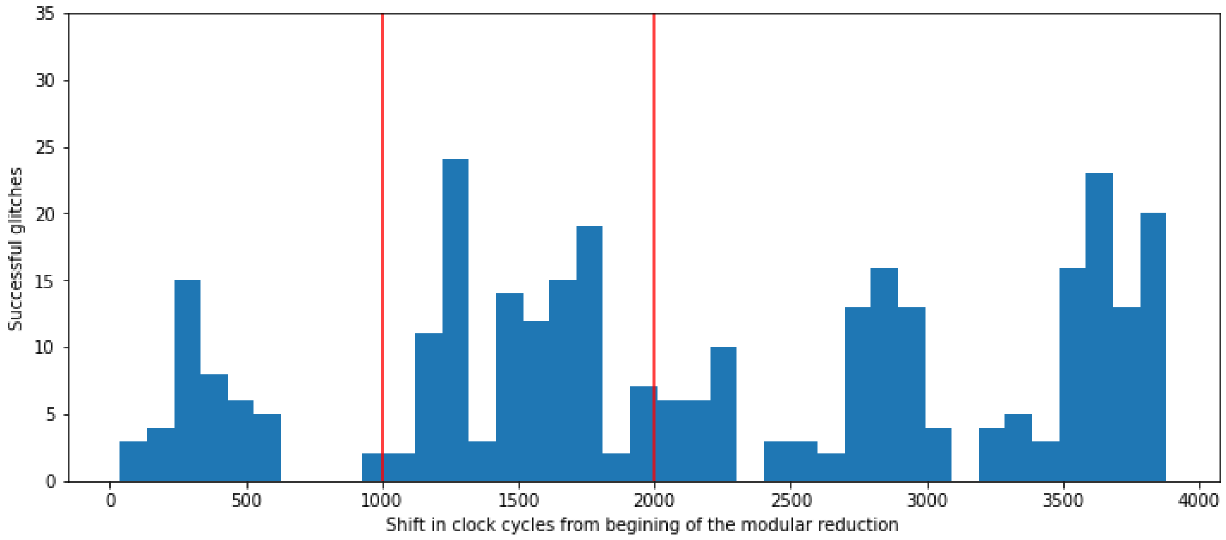

Fig. 6. Distribution of successful attacks depending on the `SHIFT` of the glitch introduction. Glitch generator parameters used: `WIDTH = 46%`, `OFFSET = −41%`, `REPEAT = 11 CLKs`, and `SHIFT = 0 − 3 930 CLKs`. Bin size is `100 SHIFTs`, where the last bin holds results for `SHIFT = 3 900 − 3 930 CLKs`. We highlighted the section `SHIFT = 1 000 − 2 000 CLKs` with the largest amount of successful glitch attacks. For a detailed explanation of parameters `WIDTH`, `OFFSET`, `REPEAT`, and `SHIFT`, please see Figure 5.

generator configurations led to successful attack and private key recovery, and we can state that there are no countermeasures protecting the processor as a whole.

It was observed that the second generator configuration had an absolute advantage in terms of successful glitches, but the first configuration had a better r_s coefficient.

B. RSA-CRT in hardware cryptographic accelerator and voltage glitching

The glitch generator was configured in the same way as it was set up for the attack on the SW implementation of RSA-CRT. The whole signing process takes 154 557 ticks and fault is injected in the first half of the signing procedure — rounded to the first 75 000 ticks — assuming that the HW accelerator does not work with both prime numbers in parallel in every run. The same hardcoded message to sign and the same key is used as in attack on SW implementation. The attack was successful and the results are presented in Table IV.

To verify whether a single generator configuration can break multiple keys, only the generator configuration with the highest r_s was used, and attack was performed on three different keys. The results are summarized in Table II and presented in more detail in Fig. 7. It can be observed that the individual absolute frequencies of successful glitch injections follow a similar trend over time when using bins of width 1000 for the distribution of glitch injection shifts.

To eliminate the possibility that this is a device-specific issue, the attack was conducted on a total of three CEC 1702 targets with the same glitch generator configuration and private key. The success rate r_s of the attack was the highest on the second device and the total number of successful glitches was the highest on the third device, as shown in Table II and in Fig. 8. Therefore, the device itself may have some influence on the glitch injection results, but not to the extent that the generator parameters need to be tailored separately for each device, as the attack was successful on all devices with total success rate above 40 % on each device.

The patterns of shifts with successful attack follow a similar trend as observed during testing with different keys in Fig. 7. Thus, this issue does not appear to be isolated to individual devices, and with the same attack setup, the MCU used does not significantly impact the results.

TABLE IV

ATTACK RESULTS ON RSA-CRT IN SW AND HW. IN BOTH ATTACKS, GLITCH SETTINGS `WIDTH = 46`, `OFFSET = -41`, AND `REPEAT = 11` WAS PROVEN TO BE MORE RELIABLE. THE RANGE OF PARAMETER `SHIFT` DIFFERS BASED ON THE IMPLEMENTATION. THE DIFFERENCE IS MORE SIGNIFICANT IN AN ATTACK ON HW IMPLEMENTATION OF RSA-CRT. FOR AN EXPLANATION OF PARAMETERS `WIDTH`, `OFFSET`, AND `REPEAT`, PLEASE SEE FIGURE 5.

WIDTH	**OFFSET**	**REPEAT**	**Success**	**Reset**	**Normal**	**Ratio r_s**	**Time**
			SW				
46	-41	11	56	96	11 060	**0.58**	18:47:40 h
48	-44	9	78	475	10 659	0.16	18:49:42 h
			HW				
46	-41	11	34 269	5 642	34 977	**6.07**	08:43:24 h
48	-44	9	47 607	17 528	9 648	2.72	15:27:20 h

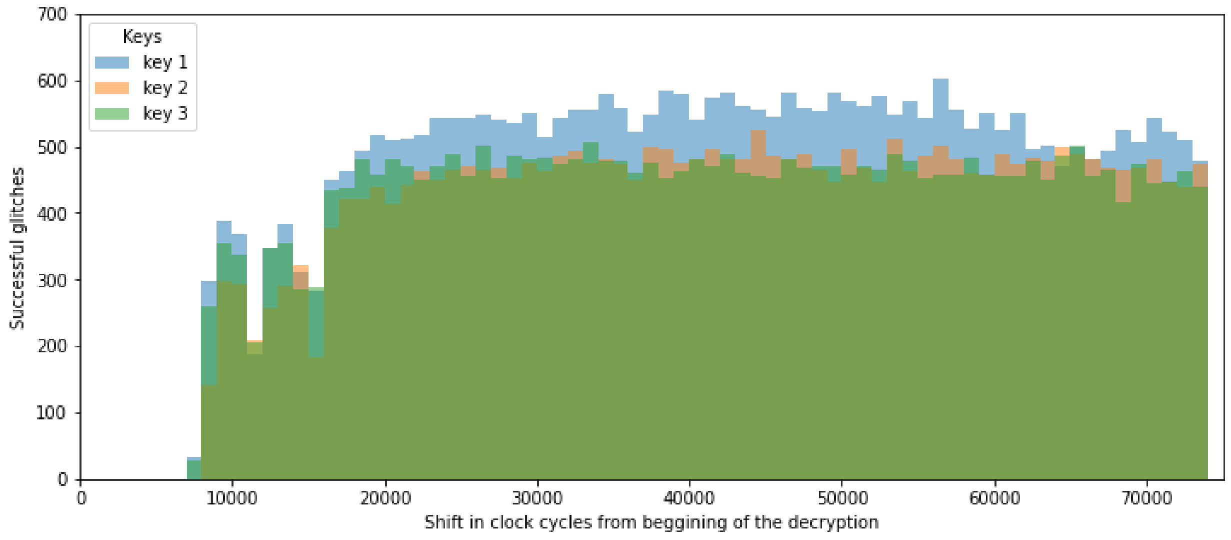

Fig. 7. Rate of successful glitches for 3 different keys using the same setting of the glitch generator: `WIDTH = 46%`, `OFFSET = -41%`, and `REPEAT = 11 CLKs`. Glitching performed for 75 000 different `SHIFTs` for each key. Bin size is `1000 SHIFTs`. For a detailed explanation of parameters `WIDTH`, `OFFSET`, `REPEAT`, and `SHIFT`, please see Figure 5.

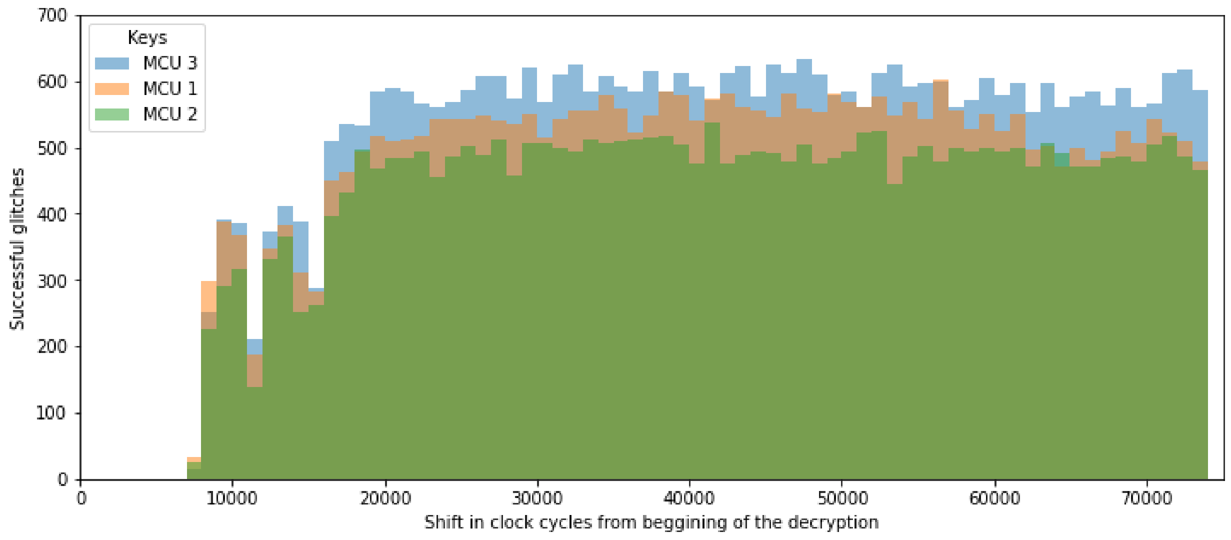

Fig. 8. Rate of successful glitches performed on 3 different MCUs with the same RSA key using the same setting of the glitch generator: `WIDTH = 46%`, `OFFSET = -41%`, and `REPEAT = 11 CLKs`. Glitching performed for 75 000 different `SHIFTs` for each MCU. Bin size is `1000 SHIFTs`. For a detailed explanation of parameters `WIDTH`, `OFFSET`, `REPEAT`, and `SHIFT`, please see Figure 5.

V. Conclusion

Fault injection attacks, particularly voltage glitching, have been demonstrated as powerful tools for compromising the cryptographic accelerator of Microchip CEC 1702 microcontroller. Bellcore/Lenstra's attack [5] on RSA-CRT showcased how intentional faults, introduced through voltage glitching, can jeopardize the security of RSA-based cryptographic systems. It also showed that even modern microcontrollers dedicated for cryptographic applications may be vulnerable to old-school attacks, although countermeasures are known and simple. Appropriate countermeasures, such as signature verification, must be then added by firmware developers.

The ChipWhisperer suite proved invaluable in executing precise fault injection attacks on both software and hardware implementations of RSA-CRT. Our findings underscore the need for enhanced security measures to prevent against fault injection attacks, especially in microcontroller-based cryptographic systems.

In conclusion, this study emphasizes the efficacy of voltage glitching as a formidable technique in the arsenal of fault injection attacks. Ongoing advancements in cybersecurity must address and mitigate these vulnerabilities to ensure the resilience of digital systems against emerging threats.

Acknowledgment

This work was supported by the Czech Technical University (CTU) grant No. SGS23/208/OHK3/3T/18 and by the Student Summer Research Program 2023 of FIT CTU in Prague.

Availability

Used implementation of RSA-CRT and created scripts utilizing ChipWhisperer environment can be found at Github repository https://github.com/CEC1702/CEC1702-RSA-CRT-Glitch.

References

[1] Agilent Technologies, Inc. Agilent technologies infiniivision 7000a series oscilloscopes[online]. https://www.keysight.com/us/en/assets/7018-08233/data-sheets-archived/5989-7736.pdf. [cit. 2024-01-12].

[2] Alessandro Barenghi, Guido Bertoni, Emanuele Parrinello, and Gerardo Pelosi. Low voltage fault attacks on the rsa cryptosystem. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 23–31. IEEE, 2009.

[3] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 513–525. Springer, 1997.

[4] Johannes Blömer and Jean-Pierre Seifert. Fault based cryptanalysis of the advanced encryption standard (aes). In *Financial Cryptography: 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003. Revised Papers 7*, pages 162–181. Springer, 2003.

[5] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[6] Christophe Clavier, Benoit Feix, Georges Gagnerot, and Mylène Roussellet. Passive and active combined attacks on aes combining fault attacks and side channel analysis. In *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 10–19. IEEE, 2010.

[7] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on aes. In *Applied Cryptography and Network Security: First International Conference, ACNS 2003, Kunming, China, October 16-19, 2003. Proceedings 1*, pages 293–306. Springer, 2003.

[8] Tereza Horníčková, Tomáš Přeučil, Martin Novotný, and Zdeněk Martinásek. Side-channel analysis of cryptographic processor cec 1702. In *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4. IEEE, 2023.

[9] Microchip Technology Inc. CEC1702 automotive cryptographic embedded controller [online]. https://ww1.microchip.com/downloads/en/DeviceDoc/CEC1702-Automotive-Data-Sheet-DS00003568B.pdf. [cit. 2022-02-25].

[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: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings 12*, pages 320–334. Springer, 2010.

[11] Neven 7 s.r.o. EEPROM flash BIOS USB programmer with CH341A SPI [online]. https://www.neven.cz/kategorie/elektronicke-soucastky/elektronicky-vyvoj/programatori/eeprom-flash-bios-usb-programator-s-cipem-ch341a-spi/. [cit. 2024-01-12].

[12] NewAE Technology Inc. Chipwhisperer overview - what is chipwhisperer? [online]. https://chipwhisperer.readthedocs.io/en/latest/getting-started.html. [cit. 2024-01-12].

[13] NewAE Technology Inc. Chipwhisperer simpleserial v1.1 [online]. https://chipwhisperer.readthedocs.io/en/latest/simpleserial.html. [cit. 2024-01-12].

[14] NewAE Technology Inc. Chipwhisperer starterkit SCAPACK-L1 [online]. https://www.newae.com/products/NAE-SCAPACK-L1. [cit. 2024-01-12].

[15] NewAE Technology Inc. Chipwhisperer version 5.5.0 [online]. https://github.com/newaetech/chipwhisperer/tree/5.5. [cit. 2022-02-25].

[16] NewAE Technology Inc. CW1173 ChipWhisperer-Lite [online]. https://rtfm.newae.com/Capture/ChipWhisperer-Lite/. [cit. 2024-01-12].

[17] NewAE Technology Inc. CW308 UFO [online]. https://rtfm.newae.com/Targets/CW308%20UFO/. [cit. 2024-01-12].

[18] NewAE Technology Inc. CW308T-CEC1702 [online]. https://rtfm.newae.com/Targets/UFO%20Targets/CW308T-CEC1702/. [cit. 2024-01-12].

[19] NewAE Technology Inc. CW308T-STM32F [online]. [cit. 2024-01-12].

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[21] Thomas Roche, Victor Lomné, and Karim Khalfallah. Combined fault and side-channel attack on protected implementations of aes. In *Smart Card Research and Advanced Applications: 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers 10*, pages 65–83. Springer, 2011.

[22] Loic Zussa, Jean-Max Dutertre, Jessy Clediere, and Bruno Robisson. Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 130–135. IEEE, 2014.