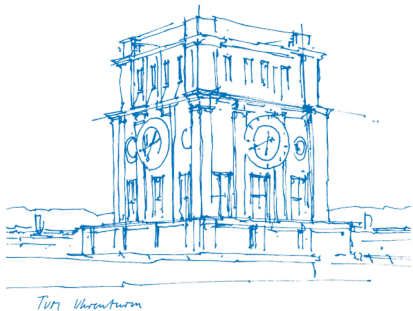
FPGA Needle: Precise Remote Fault Attacks from FPGA to CPU

Mathieu Gross[1] Jonas Krautter[2] Dennis Gnad[2] Michael Gruber[1]
Georg Sigl[1] Mehdi Tahoori [2]

[1]Technical University of Munich
Chair of Security in Information Technology
TUM School of Computation Information and Technology

[2]Karlsruhe Institute of Technology
Chair of Dependable Nano Computing

ASP-DAC, 18.01.2023



Introduction

- FPGAs are popular computation platform that are found in SoC platforms up to the cloud due to their good flexibility, computing and power efficiency
- Security is a crucial topic for FPGAs based system, especially since side-channel and fault attacks can be implemented remotely through dedicated FPGA logic

- In this work, faults attacks performed from FPGA to CPU in an FPGA-SoC context are considered

Outline

Remote Fault Injection on Software

Induce fault during the execution of software without a laboratory fault injection setup

Existing possibilities:

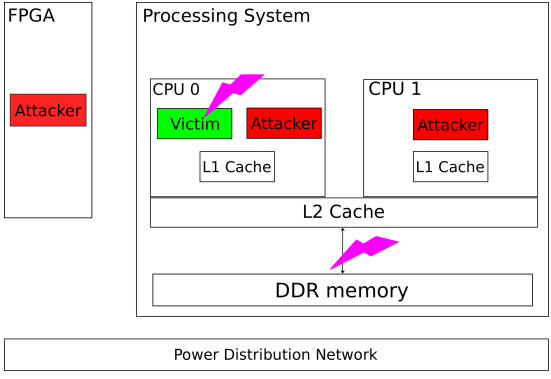
- Rowhammer [2]
- Dynamic Voltage and Frequency Scaling (DVFS) [9]
- Combine DVFS with voltage-drop generated from FPGA logic [4]

Our Solution:

- Generate faults on software via FPGA logic only

Threat Model

- Power Distribution Network shared between FPGA and CPU
- Attacker located on CPU 0 or 1 with user privileges
- Attacker can partially reconfigure the FPGA from software
- Attacker logic generate voltage drops that affect Victim's execution

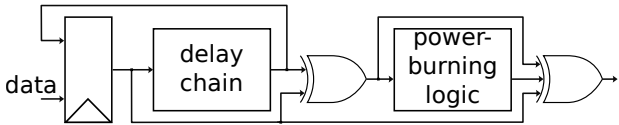


Methodology - Fault Injection through Glitch Amplifcation

Concept introduced in [5] as alternative to ring oscillators for remote fault injection in FPGAs

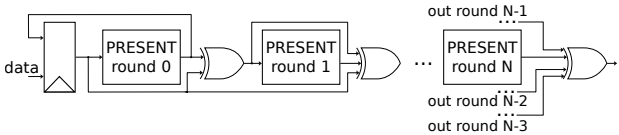
Glitch amplification circuits rely on:

- A glitch generator: Flip-flop + "delay logic" + XOR
- Power-burning network: Wires + logic along the routing path that consume dynamic power



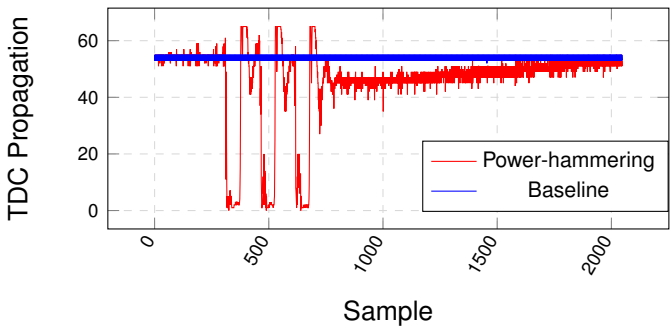
Power-hammering Circuit

- Power-hammering based on AES rounds was presented in [8]
- Power-hammering circuit implemented with PRESENT [1] rounds and XORs between rounds
- Better "voltage-drop granularity" than AES rounds



Fault injection Parameters

- Number of rounds per PRESENT power-hammer and the number of PRESENT power-hammer instances
- Activation delay offset after a trigger signal
- Total duration of the fault injection
- Period of the enable signal
- Duty-cycle of the enable signal



Experimental Setups

Platform	FPGA clock freq. (MHz)	PRESENT power -hammer (number, rounds)	Duration (cycles)	Activation freq. (MHz)	Duty cycle
Pynq-Z1	222	(13,16)	450	1.48	40 (bare-metal) 30-40 (best 31) (Linux)
Terasic DE1-SoC	250	(14,13)	10 000	0.408	99 (bare-metal)

Fault Model Evaluation

- Can instructions be skipped, executed multiple times or be faulted ?
- Is it possible to fault the data transfer from DDR to the processor's caches ?
- Is any of the knowledge obtained through those experiments exploitable for a concrete attack ?

Fault Model Evaluation - Processor Instructions

```
#define N 500
#define NUMBER_OF_NOPS 100
...
int j = 0;
/*Attacker starts injecting faults from here*/
NUMBER_OF_NOPS*nops();
j++;
...  // N consecutive j++ instructions
j++;
NUMBER_OF_NOPS*nops();
...
```

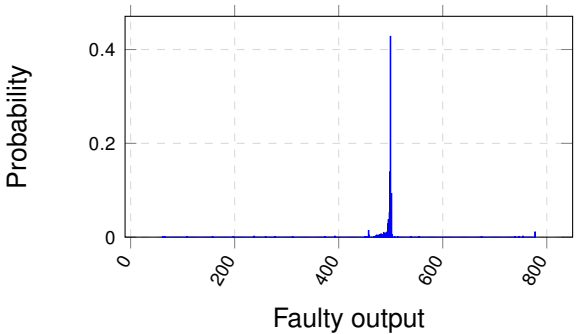
Listing 1: Faulting add instructions

```
#define N 5
#define MULTIPLIER 11
#define NUMBER_OF_NOPS 500
...
uint32_t j = 3;
// Attacker starts injecting faults from here
NUMBER_OF_NOPS*nops();
j *= MULTIPLIER;
... // N consecutive j *= MULTIPLIER
j *= MULTIPLIER;
NUMBER_OF_NOPS*nops();
...
```

Listing 2: Faulting a victim code based on a multiplication instruction

Faulty Output Distribution on the Variable Incrementation

1000 faulty outputs were collected for the Pynq-Z1 and Terasic-DE1 SoC



Faulty output range	[0-450]	[450-500[	]500-600]	[503927-504058]
Distinct faulty outputs	39	14	11	4
Faulty output distribution	150	463	23	364

Table: Faulty outputs distribution during Listing 1 execution on the Terasic DE1-SoC

Faulty Output Distribution on the Exponentiation Code

- 1000 faulty outputs were collected for the Pynq-Z1 and Terasic-DE1 SoC
- Faulty outputs classified according to the greater power of 11 divisor

Value/ $\max(11^N)$ divisor	0	3	11	11^2	11^3	11^4	11^5	11^6	others
Distinct faulty outputs	1	1	6	12	11	11	1	1	12
Faulty output distribution	141	17	29	279	314	179	1	1	39

Table: Faulty output distribution during Listing 2 execution on the Pynq-Z1

Value/ $\max(11^N)$ divisor	0	3	11	11^2	11^3	11^4	11^5	11^6	others
Distinct faulty outputs	1	0	4	1	6	7	0	0	9
Faulty output distribution	1	0	241	3	276	281	0	0	198

Table: Faulty output distribution during Listing 2 execution on the Terasic DE1-SoC

Deduced Fault Model

- Instructions skips and multiple execution of an instruction: Fault in the program counter register or processor's pipeline ?
- Faults observed on the ADD and MUL instructions
- Instruction skips has been exploited for privilege escalation [10] on an ARM-Cortex A9

Faulting Data Transfer from DDR Memory to the processor's cache

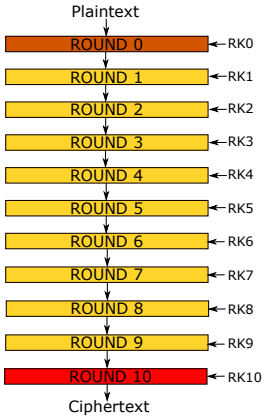
- Fault observed in consecutive words within a cache line (4 or 8 faulty words)
- Multiple types of faults observed within a word: random, single byte, multi-bytes
- Fault primitive used for implementing a fault attack on an AES T-Tables implementation

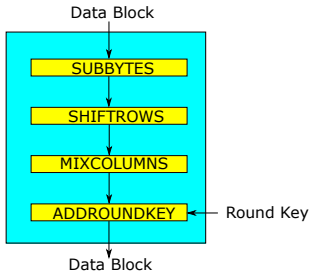
```
1  #define ARRAY_SIZE 1024
2  #define FILL_PATTERN 0xFFFFFFFF
3  ...
4  uint32_t array_attacked[ARRAY_SIZE];
5
6  fill_array(array_attacked,FILL_PATTERN);
7  flush_caches();
8  // Attacker starts injecting faults from here
9  verify_fill_pattern(array_attacked,FILL_PATTERN);
10 ...
```

Listing 3: Faulting data transfer from memory to the cache hierarchy

The AES Block Cipher

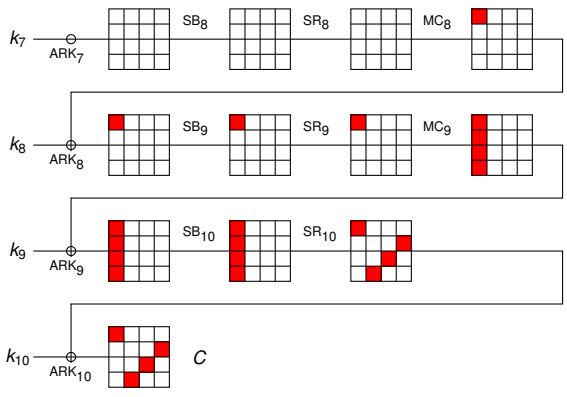
- Symetric block cipher that operates on 16 Bytes block
- Possible key size: **128**, 192, 256 bits
- A 32-bit implementation using 4 Transformation-Tables of 1 kB is used in this work (*mbedTLS* library)





Differential Fault Attack (DFA) on AES

- Based on the attack from Piret et al. [7]
- Goal: Inject a single fault between MC_8 and SB_9 that lead to a 4 bytes state difference
- Fault observed during the transfer from a T-Tables memory block (no fault observed if all the T-Tables are already cached)



Results of the DFA

- Experiments evaluated in a bare-metal scenario
- Fault injection results evaluated with 100 plaintexts, 15 configurations (each configuration is used for 10 measurements) and 10 different keys
- No T-Tables in the cache before an AES encryption

	Number of total faults	Number of exploitable faults	Ratio
Worst	595	40	6.72%
Average	620	61	9.88%
Best	607	74	12.19%

Discussion and Countermeasures

Fault attacks with Linux running is challenging because of the crashes:

- Which power-hammering configuration is optimal for avoiding crashes and injecting sufficient faults for a DFA ?
- Parameterspace explored with an automated board reset framework under a crash, but no "optimal configuration" found so far
- Future work could explore the use of reinforcement learning algorithms for parameter space exploration [6]

Possible countermeasures:

- Bitstream scanning for malicious circuit signatures [3]
- Detection of voltage drop with voltage sensors [8]: decrease the CPU clock and program a safe FPGA configuration if an attack scenario is detected

Conclusion

- Fault attacks from FPGA to CPU are possible
- The fault injection is precise enough for implementing a DFA on an AES T-Tables implementation and skip instructions
- Future work should investigate faults attacks further on a Linux setup and evaluate the effectiveness of countermeasures

Thank you for your attention!
Questions ?

References I

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe.
Present: An ultra-lightweight block cipher.
In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 450–466, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[2] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu.
Flipping bits in memory without accessing them: An experimental study of dram disturbance errors.
SIGARCH Comput. Archit. News, 42(3):361–372, jun 2014.

References II

[3] Tuan La, Khoa Pham, Joseph Powell, and Dirk Koch.
Denial-of-service on fpga-based cloud infrastructures — attack and defense.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(3):441–464, Jul. 2021.

[4] Dina Gamaleldin Ahmed Shawky Mahmoud, Samah Hussein, Vincent Lenders, and Mirjana Stojilovic.
Fpga-to-cpu undervolting attacks.
page 6, 2022.
This research is supported by armasuisse Science and Technology.

References III

[5] Kaspar Matas, Tuan Minh La, Khoa Dang Pham, and Dirk Koch.
Power-hammering through glitch amplification – attacks and mitigation.
In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 65–69, 2020.

[6] Mehrdad Moradi, Bentley James Oakes, Mustafa Saraoglu, Andrey Morozov, Klaus Janschek, and Joachim Denil.
Exploring fault parameter space using reinforcement learning-based fault injection.
In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 102–109, 2020.

References IV

[7] Gilles Piret and Jean-Jacques Quisquater.
A differential fault attack technique against SPN structures, with application to the AES and khazad.
In *Lecture Notes in Computer Science*, pages 77–88. Springer Berlin Heidelberg, 2003.

[8] George Provelengios, Daniel Holcomb, and Russell Tessier.
Mitigating voltage attacks in multi-tenant fpgas.
ACM Transactions on Reconfigurable Technology and Systems (TRETS), 14(2):1–24, 2021.

[9] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo.
CLKSCREW: Exposing the perils of Security-Oblivious energy management.
In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1057–1074, Vancouver, BC, August 2017. USENIX Association.

References V

[10] Niek Timmers and Cristofaro Mune.
Escalating privileges in linux using voltage fault injection.
In *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 1–8, 2017.